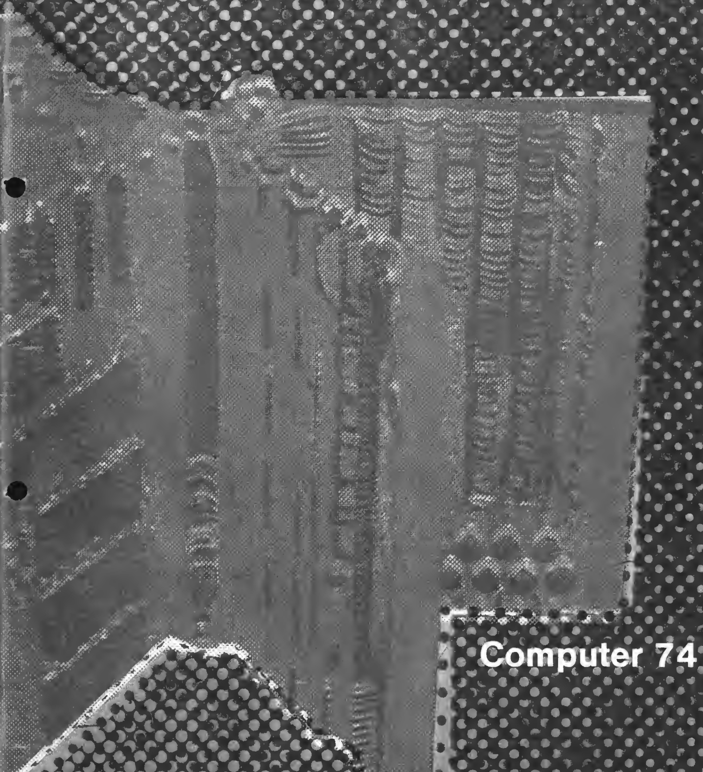
G 3078 EX

elektor

42

Fachzeitschrift für Elektronik Mai 1974 Preis DM 2,80



Computer 74

J.T.W. Damen

COMPUTER 74

Teil 1

In einer Reihe von fünf bis sechs Artikeln stellt Elektor einen Computer "für den Hausgebrauch" vor.
Mit einem Aufwand von 350 bis 2000 DM, je nach Ausbaustufe, kann ein vollwertiger 16-Bit Rechner gebaut werden. Obwohl ein Halbleiterspeicher vorgesehen ist, ist im Prinzip auch der Anschluß eines Kassettenrekorders als Speichermedium möglich.
Der Computer 74 ist sehr gut für Schulung und Demonstration geeignet, er kann aber auch auf vielen anderen Gebieten verwendet werden, die den Einsatz eines Kleinrechners erfordern.
Der Name "Computer 74" weist auf das Jahr der Entwicklung sowie auf die Verwendung von TTL-IC's der Serie 7400 hin.

Jeder Computer besteht im Prinzip aus einem Steuerwerk, einem Rechenwerk und einem Speicher.
Ein Computer führt Anweisungen oder Befehle aus, die als Programm im Speicher abgespeichert werden. Die Ausführung der Befehle erfolgt durch Datentransporte, vom Speicher zum Rechenwerk, vom Rechenwerk zum Speicher, von einer peripheren Einheit zur anderen usw..
Nach diesem Prinzip wurde auch der Computer 74 entwickelt. Durch die Ablage des Programms im Speicher ist er frei programmierbar und kann dadurch universell verwendet werden. Das unterscheidet ihn von den üblichen Taschen- und Tischrechern, die man ebenfalls, wenn auch zu unrecht, manchmal als "Computer" bezeichnet.
Der Computer 74 besteht aus einer zentralen Steuereinheit, die für Anweisungen sorgt, und aus peripheren Einheiten. Da jede dieser peripheren Einheiten eine bestimmte Funktion hat, werden sie im folgenden Funktionseinheiten genannt.
Der Computer kennt keine Befehle, sondern nur Adressen von Funktionseinheiten. Ein Auftrag wird ausgeführt, indem zwei Funktionseinheiten miteinander verbunden werden. Eine Einheit liefert die Daten, die andere empfängt und verarbeitet sie. Aufgrund der unterschiedlichen Funktion der einzelnen Einheiten ist es möglich, den Computer verschiedene Aufträge ausführen zu lassen.
Die Ausführung eines Auftrages soll nun am Beispiel einer Addition zweier Zahlen dargestellt werden. Der Steuerung stehen dazu ein Speicher und ein Addierer zur Verfügung.
Das Addieren geht dann folgendermaßen vor sich: Die Steuerung bestimmt ein Speicherwort A als SOURCE (SRC, Quelle, Lieferant) und den Addierer als DESTINATION (DST, Bestimmungsort). Danach gibt die Steuerung das Speicherwort B als SRC und wieder den Addierer als DST an. Aufgrund der Funktion des Addierers werden die beiden Werte addiert und das Resultat festgehalten. Zum Schluß wird der Addierer als SRC und das Speicherwort C als DST angegeben. Damit wird das Ergebnis der Addition der Zahlen A und B in die Speicherzelle C transportiert.

Blockschaltbild des Computers

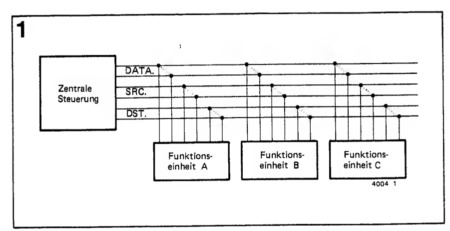
In Bild 1 ist das Blockschaltbild des Computers dargestellt.
Die Funktionseinheiten sind mit der Steuereinheit über gemeinsame Steuerleitungen verbunden. Die Steuereinheit kann über diese Steuerleitungen eine bestimmte Funktionseinheit erreichen, wenn ihr die Adresse bekannt ist. Um eine Funktionseinheit als SRC oder DST anzugeben, sind zwei Adreßleitungen erforderlich, eine SRC- und eine DST-Leitung. Für eine Datenübertragung zwischen den Funktionseinheiten wird noch eine dritte Leitung benötigt, die DATA-Leitung.
Die als SRC bestimmte Funktionseinheit gibt die in ihrem DATA-Buffer (Zwischenspeicher) vorhandene Information auf die DATA-Leitung, die DST-Funktion wertet die DATA-Leitung aus und verarbeitet die Information.

Programm

Das charakteristische Merkmal eines Computers, das ihn von einer Rechenmaschine unterscheidet, ist das im Speicher abgelegte Programm. Das Programm ist ein Plan, nach dem die Ausführung eines Auftrags erfolgt. Der Computer liest das Programm Schritt für Schritt aus, übersetzt es und setzt es in "Handlungen" (Operationen) um.
Im Computer 74 ist der Vorgang der gleiche. Im Speicher wird das Programm abgespeichert, der Computer liest es und führt es aus. Das Programm ist einfach, es besteht nur aus einer Liste von Adressen, die angeben, welche Funktionseinheiten nacheinander als SRC oder DST fungieren sollen.
Das Programm für die Addition zweier Zahlen sieht dann folgendermaßen aus:

(SRC) – Speicherzelle A
(DST) – Addierer
(SRC) – Speicherzelle B
(DST) – Addierer
(SRC) – Addierer
(DST) – Speicherzelle C

Die Speicherzellen A, B und C und der Addierer stellen dabei Adressen dar.
Wichtig ist nun, daß von Augenblick zu Augenblick festgestellt werden kann, wieweit das Programm abgearbeitet ist. Dazu dient ein spezieller Zähler: der PROGRAM COUNTER (PC). Der PC ist, wie alle anderen Funktionen, eine



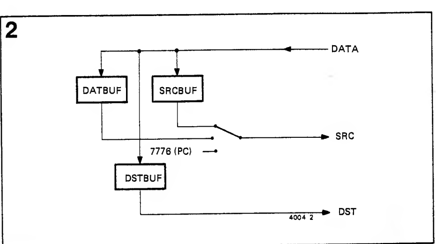
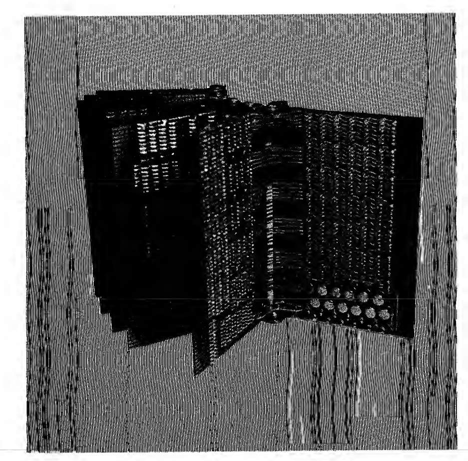


Bild 1. Blockschaltbild des "Computer 74". Die Funktionseinheiten des Computers sind für die zentrale Steuerung über Adreßleitungen erreichbar.

Bild 2. Das Prinzip der zentralen Steuereinheit des Computers.

Foto: Aufbau des Prototyps.



Einheit, die über die SRC- und DST-Leitungen erreichbar ist.
Wenn der Computer ein Programm ausführt, muß zuerst der Inhalt der PC's ausgelesen werden. Darin befindet sich die Adresse eines Speicherwortes (Startadresse). Dieses Speicherwort wird ausgelesen und die darin enthaltene Information gleichzeitig in einem Buffer der zentralen Steuerung abgespeichert. Diese Information stellt die Adresse der Funktion für den ersten SRC dar. Der PC wird mit jedem Auslesen seines Inhalts automatisch um eine Stelle erhöht. Mit dem nächsten Auslesen des PC-Inhalts wird daher die nächste Speicheradresse gefunden.
Nun wird das unter dieser Adresse stehende Speicherwort ausgelesen, der Inhalt ist die Adresse der ersten DST. SRC und DST sind nun beide bekannt, wenn sie angewiesen werden, findet die erste Datenübertragung statt. Damit ist ein Zyklus vollzogen.

Prinzip der zentralen Steuereinheit

Das Prinzip der zentralen Steuerung soll an dem bereits genannten Addierbeispiel aufgezeigt werden. Bild 2 zeigt dazu ein vereinfachtes Blockschaltbild der Steuereinheit. In Tabelle I sind alle im Beispiel vorkommenden Speicher-

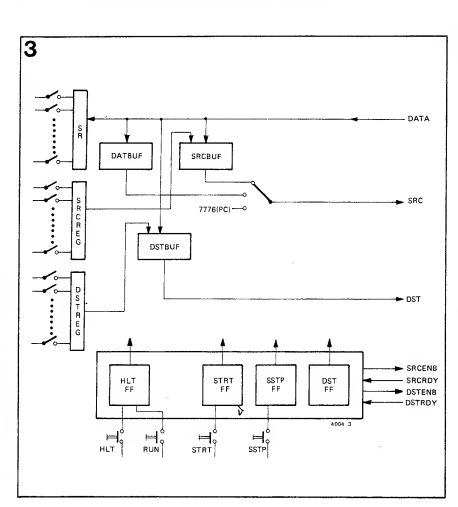


Bild 3. Blockschaltbild der zentralen Steuerung.

Bild 4. Flußdiagramm des Ablaufs der Steuereinheit.

Tabelle I

Speicher-adresse	Inhalt	Name der Speicher-adresse	Bedeutung des Inhalts
0000	0007	Anfangs-adresse	Adresse der Zahl A
0001	7706		Adr. d. Addierers
0002	0010		Adr. d. Zahl B
0003	7706		Adr. d. Addierers
0004	7706		Adr. d. Addierers
0005	0011		Adr. d. Zahl C
0006	7777		Adr. v. HALT
0007	0003	Zahl A	3
0010	0004	Zehl B	4
0011	0000	Zahl C	0

adressen mit Inhalt, Benennung und kurzer Erläuterung der Bedeutung angegeben. Die Adresse des ADDIERERS ist 7706, die von HALT (ebenfalls eine Funktionseinheit) ist 7777, der PC hat die Adresse 7776.*) HALT ist demzufolge am Ende eines Programms anzugeben.

Die zentrale Steuerung führt nacheinander folgende "Handlungen" aus:

SRC := 7776 (:= bedeutet "wird") Die Adresse des PC (=7776) wird auf die SRC-Leitung gesetzt. Der PC gibt den Inhalt seines Buffers (0000, Anfangsadresse) auf die DATA-Leitung.

DATBUF := DATA Die Data-Information (=0000) wird in einen DATA-Buffer (in der zentralen Steuerung) gegeben.

SRC := DATBUF Auf die SRC-Leitung wird 0000 gegeben und Speicherplatz 0000 als SRC angewiesen. Der Speicher reagiert mit 0007 (Inhalt des Speicherplatzes 0000).

SRCBUF := DATA Die Steuerung übernimmt diese DATA (0007) und gibt sie in den SOURCE-Buffer.

SRC := 7776 Der PC wird neu ausgelesen (0001).

DATBUF := DATA Der Inhalt des PC (0001) geht wieder in den DATA-Buffer.

SRC := DATBUF Als SRC wird 0001 angewiesen, der Inhalt hiervon (=7706) kommt auf die DATA-Leitung.

DSTBUF := DATA Dieser Inhalt wird im DSTBUF aufbewahrt, dieser befindet sich in der zentralen Steuerung. Damit sind nun SRC und DST bekannt.

SRC := SRCBUF SRC 0007 wird mit
DST := DSTBUF DST 7706 (Addierer) verbunden.

Der Inhalt des Speicherplatzes 0007 (die Zahl A = 0003) wird zum Addierer transportiert (Ende eines Zyklus).

Erweitertes Blockschaltbild

Es sind noch vier weitere Leitungen vorhanden, die ebenfalls zu allen peripheren Einheiten führen (Bild 3). Zwei davon, SRCENB (SOURCE ENABLE) und DSTENB (DESTINATION ENABLE) kommen aus der zentralen Steuerung und geben an, wann die SRC- und DST-Adressen gültig sind: Erst wenn auf der Leitung eine "1" steht, darf die gewählte SRC und DST wirksam werden.

Die beiden anderen Leitungen, SRCRDY (SOURCE READY) und DSTRDY (DESTINATION READY) sind Rückmeldungsleitungen: Die angewählte SRC meldet über SRCRDY, daß ihr Auftrag ausgeführt worden ist, das gleiche gilt für DST und DSTRDY.

Die Meldungen erfolgen auf eine Weise, die als "interlocked" bezeichnet wird. Zuerst wird SRCENB (DSTENB) gesetzt, SRCRDY (DSTRDY) meldet "1", wenn der Vorgang abgeschlossen ist. Mit dieser Rückmeldung wird SRCENB (DSTENB) zurückgesetzt, danach auch SRCRDY (DSTRDY).

Um den Computer zu starten, muß der PC auf die Anfangsadresse des Programms eingestellt werden können. Dafür sind drei "Manual Register" vorgesehen.

Diese Register bestehen aus einer Reihe von Schaltern, die beiden Stellungen eines Schalters bedeuten dabei eine

*) Die Adressierung gilt für einen Computer mit 12 Bit Wortlänge, d.h., es sind 12 SRC, 12 DST und 12 DATA-Leitungen vorhanden. Die Schreibweise der Adressen ist oktal, demnach wird ein binär codiertes Oktalsystem verwendet. Darauf wird später noch näher eingegangen.

"0" oder "1". Ein Register ist für SRC (SRCREG), eines für DST (DSTREG) und eines für DATA (SR: Switch Register). Zu jedem Register gehört auch eine Lampenreihe (LED's).

Das SR kann als SRC programmiert werden, aber auch als DST (Lampenanzeige), wogegen die Anzeigen der beiden anderen Register mit jedem SRCENB b.z.w. DSTENB die auf den Leitungen vorhandenen SRC- b.z.w. DST-Adressen anzeigen.

Weiter sind noch vier Taster vorhanden, einer für "Halt", einer für "Run", einer für "Start" und ein vierter für "Single Step". Mit jeder Betätigung des Tasters "Single Step" wird ein Programmschritt durchlaufen. Die zugehörigen Flipflops sind mit HLTFF (Halt), STRTFF (Start) und SSTPFF (Single Step) bezeichnet. Die Taste "Run" ist mit dem Rücksetzeingang des HLTFF verbunden. Das Flipflop DSTFF (Destination-FF) gibt an, ob eine Adresse im SRCBUF oder DSTBUF abgelegt werden soll.

Flußdiagramm

Der gesamte Ablauf der Steuereinheit kann in einem Flußdiagramm dargestellt werden (Bild 4). Dabei bedeutet ein Rechteck eine Operation (eine "Handlung"). Ein Parallelogramm ist das Sinnbild für eine Verzweigung, wobei die Richtung von der Antwort auf die zugehörige Frage abhängt (meist wird der Zustand einer Leitung oder eines Flipflops abgefragt).

Im Flußdiagramm kommen noch zwei neue Signalnamen vor: SRCCMD (Source-Command) und DSTCMD (Destination-Command). Dabei handelt es sich um zwei Signalleitungen, die von den einzelnen Funktionseinheiten zur Steuerung laufen. Diese Leitungen werden von Funktionseinheiten benötigt, die zwar einen Befehl (SRC oder DST) von der Steuerung übernehmen, aber selbst eine neue SRC oder DST bestimmen können. Das ist bei indirekter Adressierung der Fall. Eine genaue Beschreibung der Vorgänge auf diesen Leitungen erfolgt zu einem späteren Zeitpunkt.

Ein Programmschritt des Computers 74 besteht grundsätzlich aus einer Übertragung von Daten von einer Funktionseinheit zur anderen. Dafür werden die Adressen der beiden Funktionseinheiten benötigt.

Die Steuereinheit hat daher die Aufgabe, die Adressen der beiden Funktionseinheiten zu "besorgen" und die Datenübertragung durchzuführen. Diese Aufgabe wird durch einen Ablauf der Steuerung, wie er im Flußdiagramm dargestellt ist, erfüllt. Für die Ausführung eines Programmschritts ist also ein Ablauf der Steuerung erforderlich. Betrachtet man das Flußdiagramm, so erkennt man im wesentlichen zwei Ablaufreihen (links und rechts im Bild), die mit A und B bezeichnet sind. Um mit dem Lesen eines Flußdiagramms vertraut zu machen, soll nun der Ablauf A (auch Cyclus 1 genannt) Schritt für Schritt besprochen werden.

SRC := PC Die Adresse des PC wird auf die SRC-Leitung gegeben.

SRCENB := "1" Damit ist die SRC-Adresse gültig, die SRC (in diesem Fall der PC) gibt ihren Inhalt auf die DATA-Leitung.

SRCRDY An diesem Punkt muß auf die Rückmeldung der SRC gewartet werden. SRCRDY wird erst dann "1", wenn die SRC ihren Inhalt auf die DATA-Leitung gegeben hat.

DATBUF := DATA Damit wird der PC-Inhalt im DATBUF festgehalten.

SRCENB := "0" SRCENB wird nun zurückgesetzt, weil der SRC-Auftrag ausgeführt ist.

SRCRDY Es muß gewartet werden, bis auch SRCRDY wieder "0" ist.

SRC := DATBUF Der im DATBUF abgespeicherte PC-Inhalt wird als Adresse auf die SRC-Leitung gegeben. In dem angegebenen Adderbeispiel wäre das die Adresse der Zahl A.

SRCENB := "1" Die angewählte Speicheradresse gibt ihren Inhalt auf die DATA-Leitung.

SRCRDY Warten auf die Rückmeldung.

DSTFF Beim ersten Durchlauf des Cyclus 1 ist das DSTFF "0".

SRCBUF := DATA Der Inhalt der Speicherzelle (im Addierbeispiel die Adresse der Zahl A) wird im SRCBUF gespeichert.

SRCENB := "0" SRCENB wird zurückgenommen.

SRCRDY Warten bis SRCRDY = "0".

DSTFF := "1" Das DSTFF wird gesetzt, damit ist der erste Durchlauf des Cyclus 1 beendet.

Mit dem ersten Durchlauf Cyclus 1 wurde also die Adresse der Funktionseinheit, die als SRC fungieren soll, erhalten. Im Addierbeispiel wäre das die Adresse der Speicherzelle, die die Zahl A zum Inhalt hat.
Wie aus dem Flußdiagramm ersichtlich ist, folgt nun ein Rücksprung zum Punkt A, damit wird ein zweiter Cyclus 1 durchlaufen. Aus dem PC erhält man die nächste Speicheradresse. Da am Ende des ersten Cyclus DSTFF gesetzt wurde, wird der Inhalt dieser Speicheradresse im DSTBUF festgehalten. Diese Verzweigung hat den Namen Cyclus 2.
Am Ende von Cyclus 2 ist ein Sprung nach B angegeben. Im Ablauf B, der im Mikroprogramm mit "Fortsetzung" bezeichnet ist, erfolgt dann mit Hilfe der SRC- und DST-Adressen der erste Datentransport. Der Befehl DST := DSTBUF braucht dabei nicht gegeben zu werden, da der Inhalt des DSTBUF ständig auf der DST-Leitung "steht". Bei der Addition steht am Ende dieses Ablaufs die Zahl A im Addierer. Mit einem Sprung nach Punkt C im Flußdiagramm beginnt der nächste Ablauf der Steuereinheit.
Wenn HLTFF "0" ist (die Run-Taste wurde betätigt) läuft die Steuerung so lange "rund", bis das Programm abgearbeitet ist. Für das Additionsprogramm nach Tabelle I wären z.B. drei Steuerungsabläufe erforderlich.
Ein Programm kann durch Drücken der Starttaste und der Run-Taste gestartet werden. Wird nur die Starttaste betätigt, so besteht die Möglichkeit, einen Datentransport mit manuell eingestellten Adressen durchzuführen (SRCBUF := SRCREG; DSTBUF := DSTREG). Davon wird z.B. beim Einstellen des PC Gebrauch gemacht. Soll ein Programm in Einzelschritten abgearbeitet werden, so ist die Taste "Single Step" zu drücken. Die Steuerung läuft dann nur einmal ab und muß nach jedem Ablauf mit SSTP neu gestartet werden.

Mikroprogramm

Das Flußdiagramm kann auch auf andere Weise geschrieben werden, als eine Art Programm:

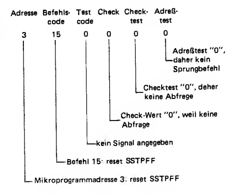
0 "START": Wenn HLTFF="0", gehe nach "BEGINN";
1 "WARTE": wenn STRTFF= "1", gehe nach "FOLGE";
2 wenn SSTPFF="0", gehe nach "WARTE";
3 reset SSTPFF;
4 "BEGINN": reset DSTFF;
5 "CYCLUS 1": gib PC auf die SRC-Leitungen;
6 set SRCENB und warte bis SRCRDY "1" wird;
7 gib DATA in DATBUF;
8 reset SRCENB und warte bis SRCRDY "0" wird;
9 gib DATBUF auf die SRC-Leitungen;
10 set SRCENB und warte bis SRCRDY "1" wird;
11 wenn DSTFF = "1", gehe nach "CYCLUS 2";
12 gib DATA in SRCBUF;
13 reset SRCENB und warte bis SRCRDY "0" wird;
14 set DSTFF;
15 gehe nach "CYCLUS 1";
16 "CYCLUS 2": gib DATA in DSTBUF;
17 reset SRCENB und warte bis SRCRDY "0" wird;
18 gehe nach "FORTSETZUNG";
19 "FOLGE": reset STRTFF;
20 wenn HLTFF = "0", gehe nach "BEGINN";
21 gib SRCREG in SRCBUF;
22 gib DSTREG in DSTBUF;
23 "FORTSETZUNG": gib SRCBUF auf die SRC-Leitungen;
24 set SRCENB;
25 "SRCWARTE": wenn SRCCMD = "1", gehe nach "SRCINDIREKT;
26 wenn SRCRDY = "0", gehe nach "SRCWARTE";
27 "SRCFORTSETZUNG": set DSTENB;
28 "DSTWARTE": wenn DSTCMD = "1", gehe nach DSTINDIREKT";
29 wenn DSTRDY = "0", gehe nach "DSTWARTE";
30 "DSTFORTSETZUNG": reset DSTENB und warte bis DSTRDY "0" wird;
31 reset SRCENB und warte bis SRCRDY "0" wird;
32 gehe nach "START";
33 "SRCINDIREKT": reset SRCENB und warte bis SRCRDY "1" wird;
34 gehe nach "SRCFORTSETZUNG";
35 "DSTINDIREKT": reset DSTENB und warte bis DSTCMD "0" wird;
36 gehe nach "DSTFORTSETZUNG".

In diesem Programm müssen 16 verschiedene Befehle ausgeführt werden können:
0 gib DATA in SRCBUF
1 gib SRCREG in SRCBUF
2 gib DATA in DATBUF
3 gib DATBUF auf SRC-Leitungen
4 gib SRCBUF auf SRC-Leitungen
5 gib PC auf SRC-Leitungen
6 set SRCENB
7 reset SRCENB
8 gib DATA in DSTBUF
9 gib DSTREG in DSTBUF
10 set DSTENB
11 reset DSTENB
12 set DSTFF
13 reset DSTFF
14 reset STRTFF
15 reset SSTPFF

Weiter müssen 8 verschiedene Signale auf "0" oder "1" abgefragt werden können:
0 SRCRDY
1 DSTRDY
2 HLTFF
3 DSTFF
4 STRTFF
5 SSTPFF
6 SRCCMD
7 DSTCMD

Außerdem muß im Programm auf folgende Adressen gesprungen werden können:
0 "START"
1 "WARTE"
4 "BEGINN"
5 "CYCLUS 1"
16 "CYCLUS 2"

Für die verschiedenen Interpretationsmöglichkeiten der kodierten Mikroprogramminstruktionen ist je ein Beispiel angegeben.



19 "FOLGE"
23 "FORTSETZUNG"
25 "SRCWARTE"
27 "SRCFORTSETZUNG"
28 "DSTWARTE"
30 "DSTFORTSETZUNG"
33 "SRCINDIREKT"
35 "DSTINDIREKT"

Das "Mikroprogramm" kann nun kodiert werden. Jede Mikroprogrammanweisung wird in eine binäre Form umgesetzt, die dann einen Befehl, eine

Tabelle II

Adresse	Befehls-code	Test-code	Check	Check-test	Adreß-test
0	4	2	0	1	1
1	19	4	1	1	1
2	1	5	0	1	1
3	15	0	0	0	0
4	13	0	0	0	0
5	5	0	0	0	0
6	6	0	1	1	0
7	2	0	0	0	0
8	7	0	0	1	0
9	3	0	0	0	0
10	6	0	1	1	0
11	16	3	1	1	1
12	0	0	0	0	0
13	7	0	0	1	0
14	12	0	0	0	0
15	5	0	0	0	1
16	8	0	0	0	0
17	7	0	0	1	0
18	23	0	0	0	1
19	14	0	0	0	0
20	4	2	0	1	1
21	1	0	0	0	0
22	9	0	0	0	0
23	4	0	0	0	0
24	6	0	0	0	0
25	33	6	1	1	1
26	25	0	0	1	1
27	10	0	0	0	0
28	35	7	1	1	1
29	28	1	0	1	1
30	11	1	0	1	0
31	7	0	0	1	0
32	0	0	0	0	1
33	7	0	1	1	0
34	27	0	0	0	1
35	11	7	0	1	0
36	30	0	0	0	1

Abfrage oder eine Sprungadresse darstellt.
Für den Befehlscode sind 4 Bit erforderlich (16 Befehle), für den Abfragecode 3 Bit (8 Abfragen) und für den Adreßcode 6 Bit (1–37 Adressen). Man würde also insgesamt 13 Bit benötigen. Aus dem Mikroprogramm geht aber hervor, daß ein Befehl und ein Sprungauftrag nie gleichzeitig vorkommen. Die 6 Bits der Sprungadresse können daher auch als Befehlscode verwendet werden, wenn durch ein zusätzliches Bit angegeben wird, um welchen Code es sich handelt. Durch zwei weitere Bits wird festgestellt, ob es sich um eine Abfrage handelt und welches Abfrageergebnis erwartet wird ("0" oder "1"). Das codierte Programm ist in Tabelle II angegeben. Befehlscode und Testcode sind dabei in dezimalen Zahlen dargestellt.
Die Interpretation einer Mikro-Instruktion ist mit Hilfe der beiden letzten Bits möglich.
Sind beide "0", so ist die Instruktion ein echter Befehl, ist der Adresstest "1" und der Checktest "0", dann muß der Code als Sprungbefehl aufgefaßt werden.
Wenn allein der Checktest "1" ist, dann muß gewartet werden, bis das unter Testcode angegebene Signal den Wert annimmt, der unter Check angegeben ist. Sind dagegen beide Bits "1", dann wird das unter Testcode angegebene Signal auf den unter Check angegebenen Wert geprüft. Ist der Wert vorhanden, folgt ein Sprung nach der Mikroprogrammadresse, die unter dem Befehlscode angegeben ist. Ist der Checkwert nicht erfüllt, dann folgt gewöhnlich die folgende Mikroprogramm-Instruktion.
Hierzu sind im Kasten unten Beispiele angegeben.
Im Teil 2 wird die "Hardware", die schaltungstechnische Realisierung des Mikroprogramms und der Steuereinheit, beschrieben.

Erklärung der Abkürzungen:

DATA (Daten, Information)
DATBUF Data Buffer
DST Destination (Bestimmung; Bestimmungsort)
DSTBUF Destination Buffer
DSTCMD Destination Command (zusätzliche Leitung zur Bestimmung einer Destination durch eine Funktionseinheit)
DSTENB Destinetion Enable (Leitung, die angibt, wann die DST-Adresse gültig ist)
DSTFF Destination Flipflop (wenn es gesetzt ist, wird eine Adresse im DSTBUF abgelegt)
DSTRDY Destination Ready (Leitung, die angibt, daß die DATA von der DST empfangen wurde)
DSTREG Destination Register (Register zur manuellen Einstellung einer DST-Adresse)
HLT Halt (Taste zum Setzen des Halt-Flipflops)
HLTFF Halt-Flipflop (wenn es gesetzt ist, wird die Steuerung am Ende eines Ablaufs angehalten)
PC Program Counter (Zähler, der mit jedem Programmschritt hochgezählt wird)
RUN (Taste zum Rücksetzen des HLT-FF)
SR Switch Register (Register zur manuellen Eingabe und Anzeige von DATA)
SRC Source (Quelle, Lieferant)
SRCBUF Source Buffer
SRCCMD Source Commend (analog DSTCMD)
SRCENB Source Enable
SRCRDY Source Reedy
SRCREG Source Register
SSTPFF Single-Step Flipflop
STRT Start (Teste zum Setzen des Start Flipflop's)
STRTFF Start-Flipflop (muß zum Sterten eines Programms gesetzt werden)

Adresse	Befehls-code	Test code	Check	Check-test	Adreß-test
15	5	0	0	0	1

- Adreßtest "1", deher Sprungbefehl
- Checktest "0", deher keine Abfrage
- Checkwert "0", weil keine Abfrege
- kein Signel engegeben
- de Adreßtest "1" ist, hendelt es sich um eine Sprungedresse, Adresse 5: "CYCLUS 1"
- Mikroprogremmadresse 15: gehe nach "CYCLUS 1"

Adresse	Befehls-code	Test-code	Ceck	Check-test	Adreß-test
6	6	0	1	1	0

- kein "Sprung"
- Abfrage
- es wird eine "1" erwertet
- Testcode 0: SCRRDY
- Befehl 6: set SRCENB
- Mikroprogrammadresse 6: set SRCENB und warte bis SRCRDY "1" wird

Adresse	Befehls-code	Test-code	Ceck	Check-test	Adreß-test
0	4	2	0	1	1

- Adreßtest "1": Sprungbefehl
- Checktest "1": Abfrege
- Checkwert "0": Abfrage euf "0"
- Testcode 2: HLTFF
- de Adreßtest "1", Adresse 4: "BEGINN"
- Mikroprogrammedresse 0: "START". Wenn HLTFF = 0, gehe nach "Beginn".

Teil II

computer 74

J.T.W. Damen

Der erste Teil des Artikels befaßte sich mit dem Prinzip des "Computer 74", dabei wurde anhand des Flußdiagramms der Steuereinheit ein Mikroprogramm entwickelt. In diesem Teil wird nun beschrieben, wie das Mikroprogramm die Wirkungsabläufe des Computers steuert.

Read Only Memory

Das gesamte Mikroprogramm ist fest verdrahtet in einem "Read Only Memory" (abgekürzt: ROM = Festwertspeicher) enthalten, dessen Prinzip in Bild 5 angedeutet ist. Der Speicherinhalt besteht aus 37 Worten (0 . . . 36) von 12 bit, davon sechs für den Befehlscode, drei für den Testcode und je eins für Check, Checktest und Adressentest. Jedes Wort setzt sich in Übereinstimmung mit dem Mikroprogramm aus binären Nullen und Einsen zusammen.

Sowohl die Adressen-Auswahl als auch der Festwertspeicher selbst sind als Diodenmatrix ausgeführt. Ein Adressen-Zähler steuert die Adressen-Auswahl, er kann im Prinzip 64 Adressen angeben, von denen eine Anzahl nicht gebraucht wird. Um zu verhindern, daß der Zähler zu bestimmten Zeitpunkten eine unbenutzte Adresse anweist, die dann unerwünschte Funktionen auslösen könnte, sind auch diese Adressen selektiert und mit einer Mikroinstruktion versehen, die das Programm auf die Adresse 0 springen läßt.

Die Gesamtschaltung des ROM's ist in Bild 6 angegeben. Jede Adressen-Auswahlleitung besteht aus einer mit sechs Germaniumdioden aufgebauten AND-Schaltung, auf die ein Inverter folgt. Nur wenn alle Eingangsleitungen "1" sind, ist der Ausgang des Inverters "0". Dieser Ausgang steuert über Dioden im Festwertspeicher einzelne der 12 Leitungen zu "0", die ihrerseits invertiert wieder eine "1" liefern.

Somit erscheint die Mikroinstruktion am Ausgang des Speichers. Die ersten sechs Leitungen des Speichers führen zu den Paralleleingängen der Binärzähler SN 74161. Erhalten die Zähler einen "load"-Befehl (LD), so wird eine neue Adresse "eingetaktet", es erfolgt ein Sprung im Mikroprogramm.

Die Mikro-Instruktionen aus dem ROM müssen nun in der richtigen Weise interpretiert werden, das geschieht mit Hilfe der ROM-Steuerung, deren Schaltung Bild 7 zeigt. Bit 10 und 11 werden in vier Gattern INC, INCTST, ADR und ADRTST dekodiert. Das erste Gatter erzeugt ein inkrementiertes Adressen-Signal (inkrementieren: um 1 erhöhen), das den ROM-Adressenzähler veranlaßt, die nächste Mikro-Instruktion auszulesen. INCTST arbeitet ebenso, aber über das Gatter CHKTST. Dieses Gatter wird aus einer

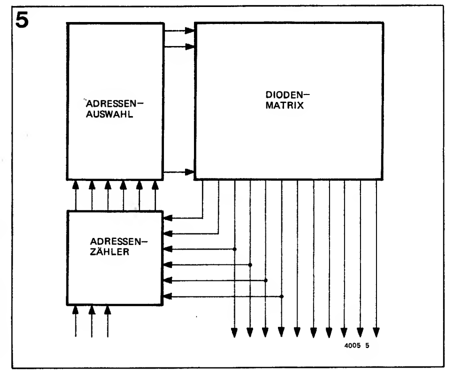
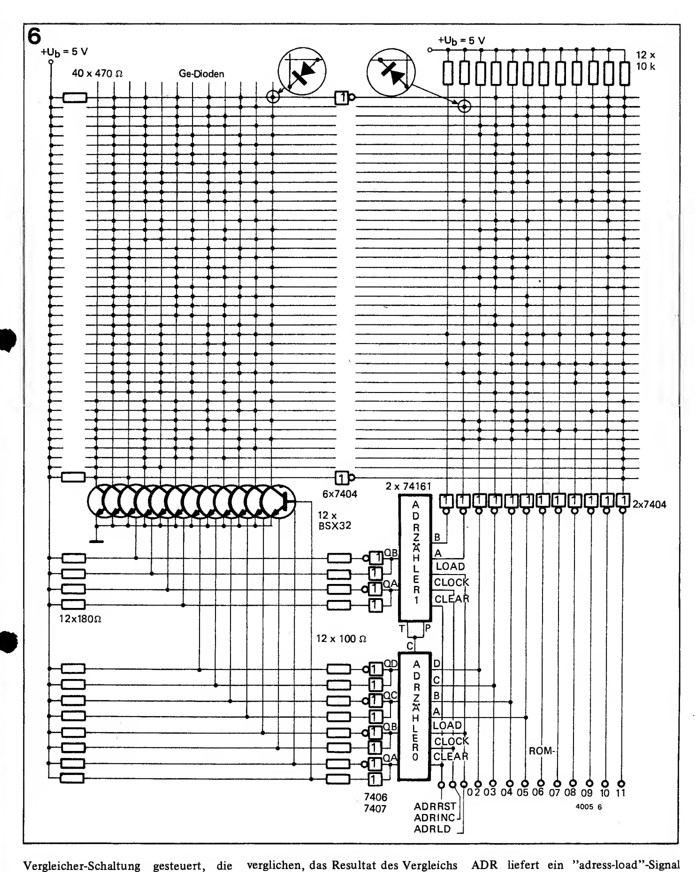


Bild 5. Des "Read Only Memory" (ROM), in dem das gesamte Mikroprogramm in fester Verdrahtung gespeichert ist.

Bild 6. Die Gesamtschaltung des Festwertspeichers (ROM). Aus der vergrößerten Detailzeichnung ist zu ersehen, deß die eus Platzmengel als Punkte eingezeichnetan Verknüpfungspunkte aus Germenium-Dioden bestehen. Jede Adressen-Auswahlleitung besteht aus einer AND-Scheltung, mit sechs Ge-Dioden und nachfolgendem Inverter.

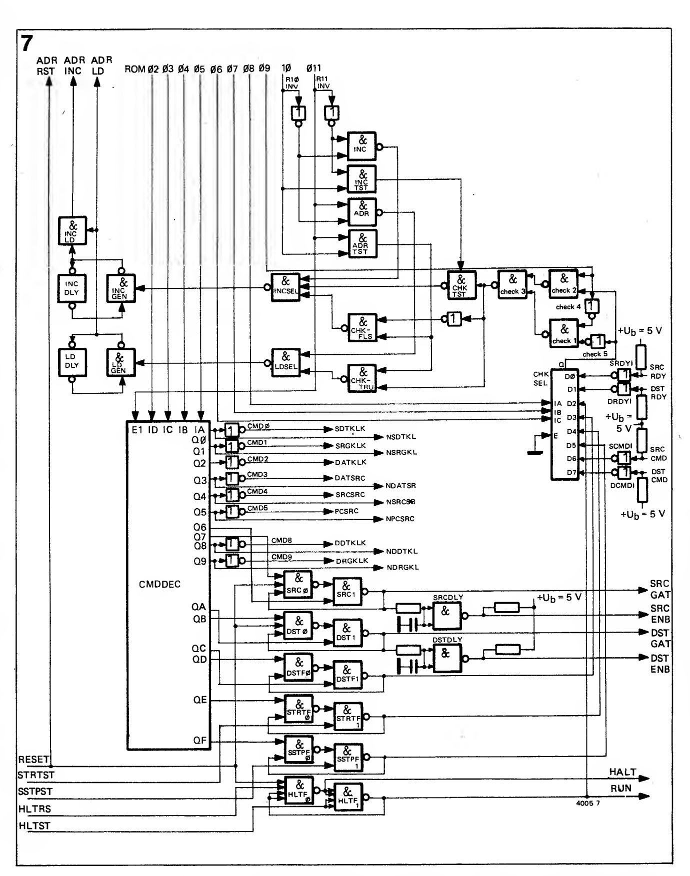


Vergleicher-Schaltung gesteuert, die aus CHECK 1, 2, 3, 4 und 5 besteht. CHKSEL wählt eine der acht Testleitungen (SRCRDY, DSTRDY, HLTFF, usw.) entsprechend der Kodierung von ROM 08, -07 und -06. Anschließend wird die gewählte Leitung mit ROM 09 verglichen, das Resultat des Vergleichs steuert CHKTST.

Es geschieht nichts, solange das selektierte Signal ungleich ROM 09 ist, nur wenn beide Signale gleich sind, folgt ein inkrementiertes Adressen-Signal wie oben beschrieben.

ADR liefert ein "adress-load"-Signal an den Adressenzähler, der die Adresse von den ROM-Ausgängen 00 . . . 05 übernimmt, das verursacht einen Sprung im Mikroprogramm.

ADRTST schließlich steuert die beiden Gatter CHKFLS und CHKTRU auf,



die das bereits erwähnte Vergleichssignal verarbeiten; und zwar CHKFLS das "Ungleich"-Signal, das in ein Adressen-Inkrement umgewandelt wird. CHKTRU übersetzt das "Gleich"-Signal in "adress-load". Damit ergeben sich Abläufe entsprechend den Spezifikationen von ROM 10 und ROM 11, wie sie in der Beschreibung des Mikroprogramms angegeben sind.
INCSEL steuert über das Gatter INCGEN ein Monoflop INCDLY, welches bewirkt, daß das gleiche Gatter INCGEN für kurze Zeit geschlossen bleibt. Damit wird sichergestellt, daß eine Mikro-Instruktion für eine bestimmte Mindestzeitdauer am Ausgang des ROM's vorhanden bleibt. Das ist

3

S R

SRCREG

DSTREG

DATBUF

SRCBUF

DSTBUF

DATA

SRC

7776(PC)

DST

HLT FF

STRT FF

SSTP FF

DST FF

SRCENB

SRCRDY

DSTENB

DSTRDY

4005 3

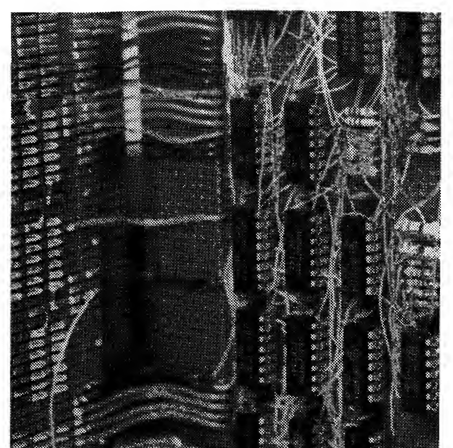
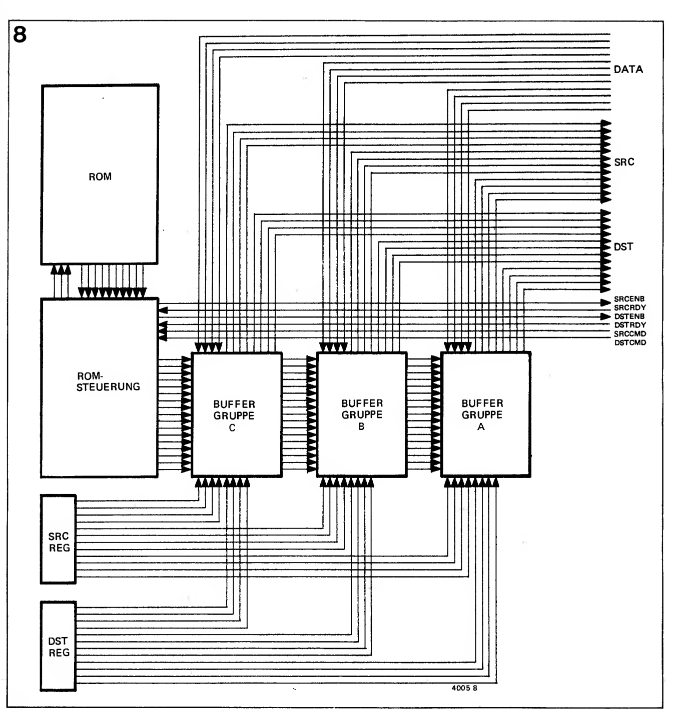


Bild 7. Die Gesamtschaltung der ROM-Steuerung. Diese Einheit bewirkt, daß die Mikro-Instruktionen aus dem ROM richtig interpretiert werden.

Bild 3. Blockschaltung der zentralen Steuerung des Computers. Dieses bereits im ersten Teil gebrachte Bild wurde hier der Verdeutlichung halber wiederholt.

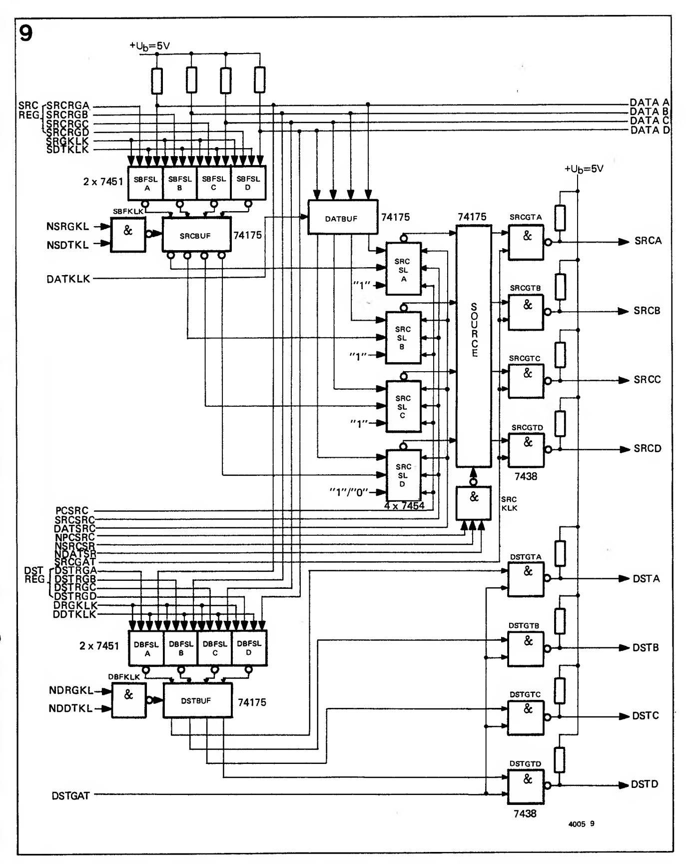
erforderlich, um die von der Instruktion ausgelösten Funktionen richtig ablaufen zu lassen. Auch bei LDSEL wird das gleiche Verfahren benutzt.
CMDDEC ist für die Ausführung der Funktionen verantwortlich, hier werden die ROM-Ausgänge ROM 02 ... 05 dekodiert, damit wird eine der sechzehn Ausgangsleitungen aktiviert. Jede Leitung entspricht einem der sechzehn Befehle. Die Dekodierung kann nur dann erfolgen, wenn ROM 11 "0" ist, denn nur in diesem Fall handelt es sich um einen Befehl. Aus diesem Grund ist der "Enable"-Eingang des SN74154 mit ROM 11 verbunden. Die sechzehn Ausgangsleitungen des 74154 - bei einigen von ihnen wird das Signal noch invertiert - führen zu den zugeordneten Steuerflipflops und zu den Buffer-



Stufen.
Die Steuerflipflops arbeiten als RS-Flipflops, sie sind aus zwei NAND-Gattern aufgebaut, FF-SRC wählt die SRC- bzw. die SRCENB-Leitung an, FF-DST die DST- bzw. die DSTENB-Leitung. Ferner sind vorhanden: Ein HALT/RUN-FF, ein DST-Flipflop, das beim Durchlauf des Mikroprogramms Verwendung findet, sowie ein START-FF und schließlich ein SSTP-FF (Single-Step-FF).

Bufferstufen

Der Computer 74 läßt sich mit einer beliebigen Anzahl von SRC-, DST- und DATA-Leitungen ausrüsten, um das zu verdeutlichen ist hier nochmals das Blockschaltbild der zentralen Steuerung aus dem ersten Teil des Artikels in Bild 3 wiedergegeben. Die verschiedenen Buffer in Bild 3 sind in Gruppen zu vier bits aufgeteilt, da viele IC's in Einheiten zu vier bit erhältlich sind. Durch die parallele Verwendung mehrerer Gruppen kann die Anzahl der bits auf 8, 12, 16 usw. erweitert werden, im Prinzip ist die Anzahl unbegrenzt.
Die Verbindungen zwischen den Buffer-Gruppen und der ROM-Steuerung gehen aus Bild 8 hervor. Die SRC-, DST-, DATA-SRCENB- und DSTENB-Leitungen beginnen bei den Buffern, während SRCRDY, DSTRDY, SRCCMD und DSTCMD über Inverter zur ROM-Steuerung führen. Die Inverter sind erforderlich, weil alle BUS-Leitungen (d.h. Leitungen, die mit den Geräten verbunden sind) bei allen Geräten eine "wired-or" Verbindung bilden, daher sind die Signale auf allen diesen Leitungen invertiert. Überall, wo eine BUS-Leitung gesteuert werden soll, geschieht das über ein Gatter mit offenem Kollektor-Ausgang.
Aus Bild 9, der Gesamtschaltung einer Buffer-Gruppe, geht hervor, daß die SRC- und DST-Leitungen von einem SN 7438 gesteuert werden, bei diesem Typ handelt es sich um ein Power-



NAND mit offenem Kollektor-Ausgang. Auch die DATA-Leitungen, SRCRDY, DSTRDY, SRCCMD und DSTCMD müssen von Ausgängen mit offenem Kollektor gesteuert werden. Auch SRCENB und DSTENB, die in erster Linie von der Steuereinheit bedient werden, sind als "wired-or" ausgeführt. Diejenigen Geräte, die indirekt adressieren sollen (über SRCCMD und DSTCMD), müssen auch diese Leitungen benutzen können.

Bild 8. Dieses Bild verdeutlicht die Verbindungen zwischen ROM, ROM-Steuerung, Registern und Buffergruppen.

Bild 9. Die Gesamtschaltung einer der Buffer-Gruppen aus Bild 8.

Erklärungen der Abkürzungen:

ADR	Adresse (leden): verursacht einen Sprung im Mikroprogramm
ADRTST	Adressentest: verursacht einen Sprung, wenn der Test (Abfrage) ein positives Resultet gebracht hat
CHK	Check: Abfrage, ob ein bestimmter Befehl ausgeführt werden muß oder nicht
CHKFLS	Check False: des ebgefragte Signel hat nicht den gewünschten Wert
CHKSEL	Check Select: Auswahl der Signalleitungen, die ebgefragt werden müssen
CHKTRU	Check True: das ebgefragte Signal het den gewünschten Wert
CHKTST	Checktest: Abfrage, ob ein Check ausgeführt werden muß
CMDDEC	Kommando-Dekoder
ADRINC	Increment (um eins erhöhen): einen Schritt in Mikroprogramm weiter gehen
INCDLY	Increment Delay: stellt sicher, daß das INC-Signal eusreichend lange ansteht
INCGEN	Increment Generetor
INCSEL	Increment Select
INCTST	Incrementtest: Increment, wenn das Resultat der zugehörigen Abfrage positiv ist
ADRLD	Load: Einlesen der Adresse der nächsten Instruktion, es erfolgt ein Sprung im Mikroprogramm
LDDLY, LDGEN, LDSEL	Erzeugen zusammen ein Load-Signal (vergleiche INC, INCDLY usw.)
ROM	Read Only Memory: Festwertspeicher, in diesem Fall durch eine Diodenmatrix realisiert.

Die Schaltung der Buffergruppe ist leicht zu durchschauen. Die Buffer aus Bild 3 werden von IC's der Typs SN 74175 gebildet, jeder dieser Bausteine enthält vier D-Flipflops. Vor SRCBUF und DSTBUF liegt eine Auswahlschaltung (2x SN 7451 oder SN 7452), die entweder die DATA-Leitungen oder die manuellen SRC- und DST-Register auswählt. Der DSTBUF steuert die DST-Leitungen, SOURCE wirkt als Zwischenbuffer, dem von drei Seiten Signale zugeführt werden können. Das ist auch aus Bild 3 zu ersehen.
Eine der drei Seiten ist die Adresse des PC, die stets aus binären Einsen besteht, abgesehen vom letzten (least significant) bit, das "0" ist. An eine der Gruppen gelangt hier also eine "0". ◄

Teil II

modulations-verfahren

Wie in Teil I bereits angekündigt, befaßt sich dieser Beitrag mit der Trägerlagemodulation (CPM), der Frequenz- sowie der Phasenmodulation. Für die UKW-Übertragung hat sich die Frequenzmodulation als das beste System bewährt, während die CPM sich optimal zur Sprachübertragung eignet.

Trägerlagemodulation (Carrier Position Modulation, CPM)

Verwendet man einen Sprach-Clipper, so sind zwei Fragen zu klären:
1. Wie hoch ist der Gewinn, der mit diesem Verfahren zu erreichen ist?
2. Inwieweit wird die Verständlichkeit beeinflußt?

Experimente mit hochfrequenten Clippern ergaben, daß die Verständlichkeit selbst bei unendlicher Begrenzung noch gut blieb, weil die mittlere Leistung um etwa 10 dB anstieg. Die Verständlichkeit ließ sich noch verbessern, wenn der NF-Übertragungskette ein Preemphasis-Glied vorangeschaltet wurde.
Auch bei unendlicher Begrenzung besitzt ein gefiltertes SSB-Signal noch Amplitudenveränderungen, da die schnellen Phasensprünge von SSB-Signalen aus dem gesendeten Frequenzspektrum herausfallen.
Soll das geklippte SSB-Signal von allen Amplitudenschwankungen befreit werden, so ist eine weitere Behandlung dieses Signals erforderlich, es liegt auf der Hand, daß eine PLL-Schaltung sich dazu eignet. Bild 16 zeigt das Prinzip einer CPM-Schaltung. Dabei wird ein SSB-Signal erzeugt, wobei das NF-Signal eine Preemphasis erhält, nach der Begrenzung wird dieses Signal dann einer PLL-Schaltung zugeführt. Der VCO schwingt dann mit der gleichen Frequenz wie das SSB-Signal, es sind aber keine Amplitudenänderungen mehr vorhanden. Die PLL ist so dimensioniert, daß sie den schnellen Phasensprüngen des SSB-Signals nicht folgen kann; deshalb vergrößert sich die Bandbreite des CPM-Signals kaum oder gar nicht gegenüber derjenigen des ursprünglichen SSB-Signals. Im Hinblick auf die Verständlichkeit läßt sich mit CPM eine beträchtliche Erhöhung des Wirkungsgrads erzielen.
Der Zusammenhang zwischen Verständlichkeit und Empfänger-Eingangsspannung für verschiedene Modulationsverfahren ist in Bild 17 aufgezeigt. Dabei wurden Wortreihen ohne jeden Zusammenhang übertragen; der ZF-Teil des Empfängers war für jedes Übertragungsverfahren optimal ausge-

Computer 74

Teil 3

In den bereits veröffentlichten Teilen der Artikelreihe "Computer 74" wurde die Arbeitsweise ausführlich besprochen, ferner wurde die Ablaufsteuerung durch ein Mikroprogramm dargestellt. Nunmehr folgen die Beschreibung des Bedienungspaneels sowie Angaben über den Aufbau.

Die SRC- und DST-Register (Handbedienung) bestehen aus einer Reihe von Schaltern, als Anzeigen können Lämpchen oder LED's Verwendung finden. Aus den Bildern 8 und 10 ist u.a. ersichtlich, daß die Schalter direkt mit den zugehörigen Eingängen der Buffer-Gruppen verbunden sind. Die Selektion dort sorgt dafür, daß die im Register vorhandene Information im richtigen Moment übernommen wird.
Bei den Displays verhält es sich etwas anders. Sie erhalten die Informationen über die SRC- und DST-Leitungen, die Informationen sind aber nur für eine kurze Zeitdauer vorhanden. Um, besonders bei "single step", sehen zu können, was geschieht, ist ein Zwischenspeicher vorgesehen. Bild 10 zeigt, daß für jeweils 4 bit ein SN 74 175 als Speicher dient, in den die Signale mittels SRCENB oder DSTENB "eingetaktet" werden. Die Speicherausgänge steuern die LED's (oder Lämpchen).
Wiederum andere Verhältnisse gelten für SR (Switch Register für DATA), dieses Register ist stets über das Programm zu erreichen. Als SRC muß die von den Schaltern stammende Information auf die DATA-Leitungen gelangen, als DST hingegen muß die auf den DATA-Leitungen vorhandene Information durch die LED's angezeigt werden. Es muß daher eine Adressen-Auswahl stattfinden. Die Bilder 11 und 12 zeigen das SR in den Funktionen SRC bzw. DST.

Adressenauswahl

Die Adressen-Auswahl hat für alle "peripheren" Einheiten die gleiche Struktur. Die SRC- oder DST-Leitungen steuern diese Einheit über Inverter oder über AND-Gatter, die Wahl einer peripheren Einheit als SRC oder DST geschieht getrennt. Da die Signale auf den BUS-Leitungen invertiert sind, laufen diejenigen Leitungen, die in der Adresse "1" sein müssen, über Inverter.
Aus Bild 12 ist zu entnehmen, daß die Adresse vom SR (die DST-Leitungen; untere linke Ecke, von oben nach unten gelesen) 111.111.111.100 = 7774 lautet. Diese Information wird zusammen mit SRCENB oder DSTENB erneut AND-verknüpft, womit dann ein "address-selected"-Signal (SEL) verfügbar ist. Anschließend folgt eine mit DEVSEL (device-select) bezeichnete Schaltungsgruppe, deren Aufbau Bild 13 zeigt. Sie enthält ein RS-Flipflop, das mit der positiven Flanke des SEL-Signals gesetzt wird, Reset erfolgt mit der negativ gerichteten Flanke von ENB (SRC- oder DSTENB), also dann, wenn diese Leitung auf "0" geht. Der Ausgang des RS-Flipflops aktiviert die Einheit. Gleichzeitig gelangt die Ausgangsinformation des RS-Flipflops, verzögert durch ein RC-Glied, als RDY-Signal (SRC- oder DSTRDY) an BUS.
Die Arbeitsweise des Flankendetektors (in Bild 13) ist aus Bild 14 ersichtlich. Das Signal wird dem einen Eingang eines NAND-Gatters direkt, dem anderen Eingang aber verzögert und invertiert zugeführt. Ist das Signal "1" ($> +3$ V), so ist ein NAND-Eingang "1", der andere "0", demzufolge ist der NAND-Ausgang "1". Wird das Eingangsignal anschließend "0" (0 V), so wird der erste Eingang unmittelbar "0"; hingegen verbleibt der andere Eingang wegen der Signalverzögerung (RC-Glied!) noch kurze Zeit im Zustand "0" bevor er "1" wird. Somit ist zu jeder Zeit mindestens ein NAND-Eingang "0", der Ausgang bleibt "1": das NAND "sieht" die negative Flanke nicht! Geht das Signal nun wieder auf "1", so herrscht dieser Zustand auch sofort am ersten NAND-Eingang, der zweite Eingang verbleibt noch kurz auf "1" um dann "0" zu werden. Somit wird der NAND-Ausgang für kurze Zeit "0". Die Dauer des Ausgangsimpulses läßt sich mit Hilfe des vor dem Inverter angeordneten RC-Gliedes einstellen. Eine negativ gerichtete Impulsflanke läßt sich auf zwei Arten detektieren: Entweder nach der oben geschilderten Methode, aber mit Hilfe eines OR-Gatters,

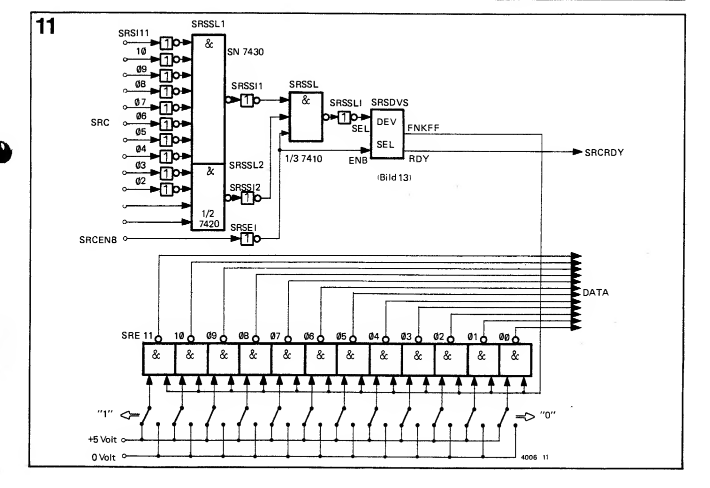
Bild 10: SRC-Register (identisch mit DST-Register) und Display. Mit den Schaltern kann aina Adresse für SRC (oder DST) einprogrammiert werden. Die ausgehende Leitungen gehen in drei Gruppen zu vier Laitungen zu den drei Buffer-Gruppen (vgl. Bild 8 und Bild 9 in Tail II). Das Displey zeigt stets die zuletzt benutzte SRC-Adresse (oder DST-Adresse) an.

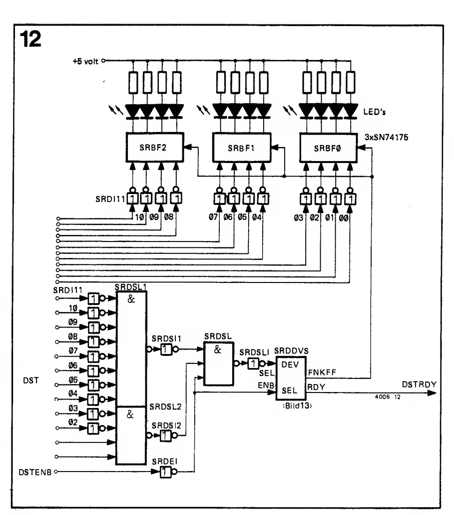
Bild 11: Das Switch-Register (SR) als SRC. Hier wird auf die glaiche Waise wie beim SRC-Register DATA einprogrammiert, zusätzlich ist hier ein Adrassen-Detaktor vorhanden, so daß DATA arst auf dan Laitungen erschaint, wann dia zentrale Stauareinheit den Befehl gibt.

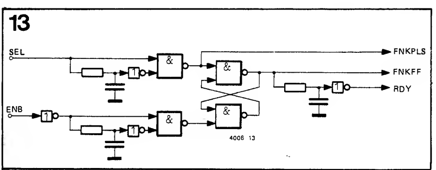
10

+5 Volt
LED's
SBF2/DBF2
SBF1/DBF1
SBFØ/DBFØ
3x SN74175
SRCENB/DSTENB
SETINV
SINV/DINV
2x SN7404
B A 9 8 7 6 5 4 3 2 1 Ø
SRC/DST
SRCREG/DSTREG
"1"
"0"
+5 Volt
ØVolt
4006 10

11







oder einfacher, durch Invertieren des Signals. Soll die Schaltung auf positiv gerichtete Impulse ansprechen (anstatt auf negativ gerichtete), so ist das NAND gegen ein AND und das OR gegen ein NOR auszutauschen.
Arbeitet das in Bild 11 gezeigte SR mit der Funktion SRC, so öffnet es eine Anzahl von Leistungs-NAND's mit offenem Kollektor, sie übertragen die Schalterstellungen (invertiert!) auf die DATA-Leitungen. Bei SR als DST (Bild 12) dient die positiv gerichtete Impulsflanke, die das RS-Flipflop setzt, gleichzeitig auch dazu, die Informationen auf den DATA-Leitungen in den Zwischenspeicher für das Display einzutakten.
Auf dem Paneel befinden sich schließlich noch vier Tastenschalter mit den Funktionen "HALT", "RUN", "START" und "SINGLE STEP". HALT und RUN steuern eine LED, die den Zustand des HALT-Flipflops anzeigt. Mit Hilfe der Drucktaster kann der Computer auf drei verschiedene Arten in Betrieb gesetzt werden (vergl. dazu auch Flußdiagramm, Teil I):
1) Taster HALT betätigen; die Schalter von SRCREG, DSTREG und SR einstellen und dann START betätigen. Es läuft ein Zyklus ab, beide Register werden gebraucht, um SRC und DST zu bestimmen.
Auf diese Weise können Instruktionen "von Hand" ausgeführt werden. Dieser Arbeitsgang wird auch benutzt, um die Anfangsadresse in PC einzubringen. Dann muß in SRCREG die Adresse des SR (7774) gesetzt werden, in DSTREG die von PC (7776) und in SR die Anfangsadresse des Programms.
2) HALT betätigen, SINGLE STEP betätigen: der Computer führt einen Schritt des eingespeicherten Programms aus. Die dabei ausgewählten SRC und DST sowie die transportierten Daten sind an den Displays abzulesen.
3) RUN betätigen, vorläufig geschieht nichts! START betätigen, das Programm läuft normal ab, es beginnt mit der Adresse in PC. Wird HALT während des Programmablaufs betätigt, so wird der gerade laufende Zyklus abgewickelt, dann geht das Mikroprogramm in den Wartezyklus.
Aus dem Flußdiagramm ist außerdem ersichtlich, daß im Fall 3) eine Betätigung von SINGLE STEP anstatt START die gleiche Wirkung hat. Falls dies unerwünscht sein sollte, so läßt es sich durch geschickte Wahl und Kopplung der Drucktasten vermeiden.
Wenn auch SRCENB und DSTENB mittels LED's angezeigt werden, lassen sich Fehler im Programm leichter verfolgen, besonders dann, wenn der Computer mitten in einem Zyklus "hängen bleibt". Das kann geschehen, wenn der Computer versucht, eine nicht bestehende Funktionseinheit auszuwählen, und anschließend auf SRCRDY oder DSTRDY wartet. Die SRCREG- und DSTREG-Displays lassen dann erkennen, welche Einheit ausgewählt wurde.

HALT

Die Funktionseinheit "HALT" ist nicht hundertprozentig dem Paneel zuzuordnen, es besteht aber eine direkte Verbindung mit der zentralen Steuereinheit. "HALT" dient ausschließlich als SRC, von dieser Einheit wird ein RESET-Signal erzeugt. Dieses Signal löst drei Dinge aus: Es setzt das HLTFF in den HALT-Zustand; bewirkt Reset der SRC- und DST-Flipflops, welche die Ausgänge der SRC- und DST-Buffer steuern; ferner erfolgt Reset der Adressenzähler des ROM, damit wird das Mikroprogramm in den Haltzyklus versetzt. Das ist der Ruhezustand des Computers.
Die Schaltung der Funktionseinheit HALT ist in Bild 15 angegeben. Der Ausgang von HALT, die Reset-Leitung, ist auch "von außen zugänglich". Sie kann über einen Drucktaster (General Reset) über ein Verzögerungsglied oder unverzögert mit der Speisespannung verbunden werden. Auf diese Weise entsteht automatisch ein Reset-Signal, wenn der Computer eingeschaltet wird.

Nummerierung der Einheiten

Bei der Auslegung des Computers wur-

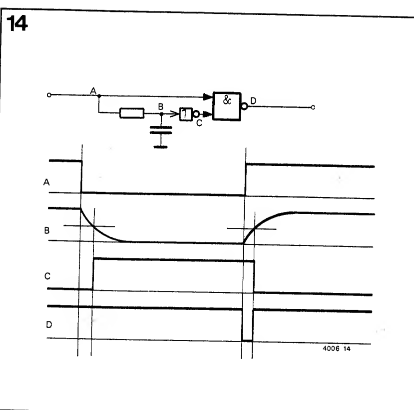
Bild 12: Das Switch-Register als DST. Die Zustände auf den DATA-Leitungen werden erst vom Display angezeigt, wenn dieses Register als DST durch die zentrale Steuereinheit angewiesen wird.

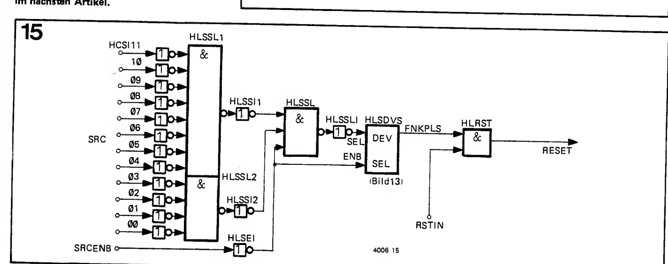
Bild 13: Device Select (DEVSEL) wird in verschiedenen Schaltbildern als ein Block angegeben (Bilder 11, 12, 15). Der so bezeichnete Block enthält dann die hier gezeigte Schaltung.

Bild 14: Prinzip des Flankendetektors, wie es bei DEVSEL angewendet wird. Das Impulsdiagramm zeigt, daß nur dann ein Ausgangsimpuls erscheint, wenn der Eingang von "0" auf "1" geht.

Bild 15: HALT ist auch eine "periphere" Einheit. Sie wird ausschließlich als SRC angewiesen und verursacht dann ein Reset-Signal für die HLT-, SRC- und DST-Flipflops sowie für den Adressenzähler im ROM.

Die Erklärungen der in den Bildern 11, 12 und 15 verwendeten Abkürzungen erfolgt im nächsten Artikel.





de ein "Standard-Satz" von Adressen für die Funktionseinheiten reserviert. Dabei hängt es vom vorgesehenen Ausbauzustand des Computers ab, ob wirklich alle Adressen nötig sind. Ausgangspunkt ist ein Computer mit 12 SRC-, 12 DST- und 12 DATA-Linien, damit können 4096 verschiedene Einheiten als SRC und als DST adressiert werden. Vorgesehen ist ein Speicher mit einer Kapazität von 1 k (= 1024) Wörtern zu 12 bit, erweiterungsfähig auf 4k – 64 (= 4096 – 64) = 4032 Wörter zu 12 bit. Die letzten 64 Adressen sind anderen Einheiten zugeordnet, sie sind in Tabelle I aufgelistet.
In dieser Tabelle gehören alle Adressen zwischen 7757 und 7700 zu einer Einheit, die 48 (oktal 60) verschiedene Funktionen mit Hilfe von "arithmetic logic units" (ALU's) des Typs SN 74 181 ausführen kann. In der Funktionsbeschreibung sind "+" und "•" die logische OR- und AND-Funktion," – " ist die invertierte Form und "x" die Exclusiv-OR-Funktion. "Plus" und "minus" sind die arithmetischen Arbeitsgänge Addieren und Subtrahieren. Die letzten sechs bits der Adresse sind faktisch die (ggf. invertierten) Befehle für die Steuereingänge M, C_n, $\overline{S}_3$, $\overline{S}_2$, S_1 und $\overline{S}_0$ des 74 181. Die letzte Adresse (111.111.)000.000 lautet somit: M = "0", C_n = "0", $S_3 = S_2 = S_1 = S_0$ = "1" (invertiert!), das entspricht der Funktion F: = a minus 1, oder in diesem Fall: RR: = RR minus 1.
Die übrigen Adressen gehören zu Hilfsregistern oder Funktionseinheiten wie Multiplizieren, Dividieren, Halt, Stack und Indirekt. Auf diese Einheiten wird später eingegangen.

Mechanische Konstruktion

Obwohl der mechanische Aufbau auf unterschiedliche Arten möglich ist, fällt stets eine Tatsache besonders auf: Alle Einheiten sind mittels einer großen Anzahl durchlaufender Drähte untereinander verbunden: Bezeichnung BUS. Damit bietet sich im Prinzip die Möglichkeit an, jede Funktionseinheit steckbar zu gestalten und für jede Platine eine Kontaktleiste mit der entsprechenden Anzahl von Kontakten vorzusehen. Wegen der großen Anzahl der benötigten Kontaktleisten wird die ganze An-

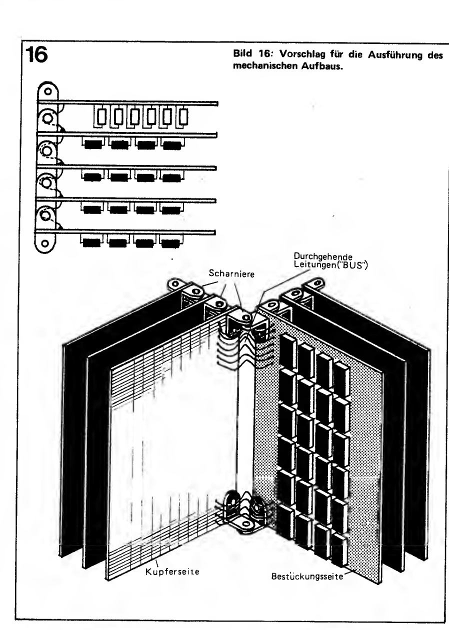
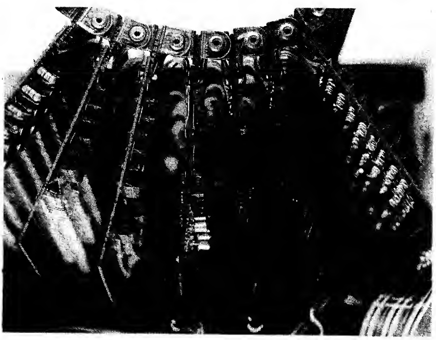


Bild 16: Vorschlag für die Ausführung des mechanischen Aufbaus.



gelegenheit aber nicht ganz billig, deshalb wurde nach einer anderen Lösung gesucht. Sie ist in Bild 16 dargestellt, hier sind die Platinen mit Hilfe von Scharnieren aneinander befestigt, etwa wie die Seiten eines Buches. Die BUS-Leitungen befinden sich an der Rückenkante dieses "Buches", die Leitungen haben ausreichend Spielraum, so daß man die "Seiten" des "Buches" umblättern kann.
Als Platinen wurden handelsübliche Ausführungen mit 60 Kupferbahnen im Abstand von 0,1 Zoll (2,54 mm) verwendet, die Bohrungen weisen den gleichen Abstand auf. Entlang eines Randes entfallen die Bohrungen, die "glatten" Leiterbahnstreifen dienen als Gegenkontakte für eine aufschiebbare Kontaktleiste, über die Verbindungen nach "außen" (z.B. zum Bedienungspaneel) hergestellt werden. Die restlichen Kupferbahnen sind durch Unterbrechungen so aufgeteilt, daß sich DIL-IC's montieren lassen. Es werden keine fertigen IC-Fassungen verwendet, sondern die im Handel erhältlichen Kontakte in Streifenform ("Gabelfederkontakte", neuerdings auch vergoldet). Die Verdrahtung verläuft auf der Bestückungsseite, sie wird zwischen den IC-Füßen hindurchgeführt, so daß nur an der hinteren Kante gelötet wird.

Zusammenfassung

In den voraufgegangenen Abschnitten wurde das Herz des Computers, die zentrale Steuereinheit beschrieben. Sie arbeitet wie eine Telefonzentrale, die zwei Einheiten miteinander verbindet. Dabei liefert eine Einheit Daten an die andere. Jede Einheit hat eine spezifische Funktion, ein Addierer zählt Zahlen zusammen, ein Speicher dient dazu, Informationen aufzubewahren, damit sie jederzeit wieder verfügbar sind, usw. Der Benutzer des Computers, der Programmierer, bestimmt, welche Einheiten miteinander in Verbindung treten müssen. Er erstellt eine Liste mit den Einheiten-Nummern, wobei jeweils zwei zusammengehören. Diese Liste, das Programm, gibt er in den Speicher ein, er "sagt" dann dem Computer, wo das Programm zu finden ist. Das heißt: Der Benutzer gibt die Speicheradresse des Programmbeginns in den Programmzähler ein.
Wird nun der Computer gestartet, so erfährt die zentrale Steuereinheit durch Abfrage des Programmzählers, womit sie beginnen soll. Dann liest die Steuereinheit jedesmal zwei Adressen aus dem Speicher aus und weist die beiden zugehörigen Einheiten als Kommunikationspartner an. Alsdann läuft das Programm ab, bis die zentrale Steuereinheit ein HALT ausliest und zur Ausführung bringt. Das Programm stoppt, der Benutzer kann den Computer erneut mit

Aufgaben befassen.

In den vorangegangenen Abschnitten wurden alle Baugruppen behandelt, die Steuerungsaufgaben zu erfüllen haben, die also dem Benutzer den "Zugang" zum Computer verschaffen. Die zentrale Steuerung ist als "Hardware-Programm" ausgeführt, das hat übrigens nichts mit dem Anwender-Programm zu tun. Letzteres steht in den Speichern, während das erstgenannte, das Mikroprogramm, unauswechselbar und fest verdrahtet, aus einer Dioden-Matrix besteht. Zwischenspeicher: Die Buffer (eine Art Privat-Speicher für die Steuerung), werden von der Steuereinheit benutzt, um alle Adressen zum richtigen Zeitpunkt an den richtigen Platz steuern zu können.

Der Benutzer hat die Möglichkeit, mit Hilfe von vier Drucktastern am Bedienungspaneel einen bestimmten Weg durch das Mikroprogramm von außen einzustellen. Eine dieser Möglichkeiten ist SINGLE STEP, dann wird das Benutzer-Programm Schritt für Schritt abgearbeitet.

In einem anderen Fall, HALT/START, sucht das Programm die Adressen von SRC und DST nicht im Speicher, sondern fragt sie an den Schaltern am Paneel (SRCREG und DSTREG) ab. Somit kann ein SRC und DST von Hand angewiesen werden. Da am Paneel noch eine Reihe von Schaltern vorhanden ist, die als SRC programmiert werden können (Switch Register), lassen sich von Hand Daten eingeben und zu allen Einheiten übertragen, die die Steuereinheit als DST anweisen kann. Eine dieser Einheiten ist der Programm-Zähler, der sich über diesen Weg auf die Startadresse des Anwender-Programms einstellen läßt. Auch das Programm selbst kann auf diese Weise eingespeichert werden.

Das "Herz" des Computers wird von den folgenden Baugruppen gebildet:
Read-Only-Memory (ROM) mit ROM-Steuerung (Mikro-Programm und zentrale Steuerung)
Buffer-Gruppen (Privat-Speicher)
Sourceregister und Display (von Hand SRC anbieten; SRC auslesen)
Destination-Register und Display (von Hand DST anbieten; DST auslesen)
HALT (stoppt die Ausführung eines Programms).

Im Grunde genommen gehört das Switch-Register nicht in diese Aufzählung: es ist eine vollständige periphere Einheit (Schalter als SRC, Display als

Tabelle I

Binäre Adresse	oktal	Abkz.	Funktion
111 111 111 111	7777	HL	halt
111 111 111 110	7776	PC	program counter (Programm-Zähler)
111 111 111 101	7775	SP	stack pointer
111 111 111 100	7774	SR	switch register und display
111 111 111 011	7773	DE	destination indirekt register
111 111 111 010	7772	SC	source indirekt register
111 111 111 001	7771	ST	stack
111 111 111 000	7770	ID	indirekt
111 111 110 111	7767	R2	Multiplizier-/Dividierregister
111 111 110 110	7766	GL	Gleichheitsregister
111 111 110 101	7765		
111 111 110 100	7764	CO	carry out register
111 111 110 011	7763	R1	Rechenregister 1
111 111 110 010	7762	R0	Rechenregister 0
111 111 110 001	7761	DL	Dividieren
111 111 110 000	7760	VM	Multiplizieren
111 111 101 111	7757		RR: = —RR (RR = R1 und R0)
111 111 101 110	7756	NOR	RR: = —(RR + DATA)
111 111 101 101	7755		RR: = —RR. DATA
111 111 101 100	7754	ZRO	RR: = 0
111 111 101 011	7753	NND	RR: = —(RR. DATA)
111 111 101 010	7752	NOT	RR: = —DATA
111 111 101 001	7751	XOR	RR: = RR∗DATA (exclusive or)
111 111 101 000	7750	MSK	RR: = RR. —DATA
111 111 100 111	7747		RR: = —RR + DATA
111 111 100 110	7746	NXR	RR: = —(RR∗DATA) not exclusive or)
111 111 100 101	7745	RR	RR: = DATA (gleich R0)
111 111 100 100	7744	AND	RR: = RR. DATA
111 111 100 011	7743		ONE RR: = 1
111 111 100 010	7742		RR: = RR + —DATA
111 111 100 001	7741	OR	RR: = RR + DATA
111 111 100 000	7740		RR: = RR
111 111 011 111	7737	INC	RR: = RR plus 1
111 111 011 110	7736		RR: = (RR + DATA) plus 1
111 111 011 101	7735		RR: = (RR + —DATA) plus 1
111 111 011 100	7734	ZRO	RR: = zero
111 111 011 011	7733		RR: = RR plus (RR. —DATA) plus 1
111 111 011 010	7732		RR: = (RR + DATA) plus (RR. —DATA) plus 1
111 111 011 001	7731	AF	RR: = RR minus DATA
111 111 011 000	7730		RR: RR. —DATA
111 111 010 111	7727		RR: = RR plus (RR. DATA) plus 1
111 111 010 110	7726		RR: = RR plus DATA plus 1
111 111 010 101	7725		RR: = (RR + —DATA) plus (RR. DATA) plus 1
111 111 010 100	7724		RR: = RR. DATA
111 111 010 011	7723		RR: = RR plus RR plus 1
111 111 010 010	7722		RR: = (RR + DATA) plus RR plus 1
111 111 010 001	7721		RR: = (RR + —DATA) plus RR plus 1
111 111 010 000	7720		RR: = RR
111 111 001 111	7717		RR: = RR
111 111 001 110	7716		RR: = RR + DATA
111 111 001 101	7715		RR: = RR + —DATA
111 111 001 100	7714	MIN	RR: = minus 1 (2er complement)
111 111 001 011	7713		RR: = RR plus (RR. —DATA)
111 111 001 010	7712		RR: = (RR + DATA) plus (RR. —DATA)
111 111 001 001	7711		RR: = RR minus DATA minus 1
111 111 001 000	7710		RR: = (RR. —DATA) minus 1
111 111 000 111	7707		RR: = RR plus (RR. DATA)
111 111 000 110	7706	OP	RR: = RR plus DATA
111 111 000 101	7705		RR: = (RR + —DATA) plus (RR. DATA)
111 111 000 100	7704		RR: = (RR. DATA) minus 1
111 111 000 011	7703	ROT	RR: = RR plus RR (Schiebe 1 nach links)
111 111 000 010	7702		RR: = (RR + DATA) plus RR
111 111 000 001	7701		RR: = (RR + —DATA) plus RR
111 111 000 000	7700	DEC	RR: = RR minus 1

Tabelle I: Adressenzuteilung für alle peripheren Einheiten, ausgenommen Speicher. Die Adressen 7757 ... 7700 sind alle der mit ALU's aufgebauten Recheneinheit zugeordnet, sie führt die zugehörige Funktion abhängig von der verwendeten Adresse aus.

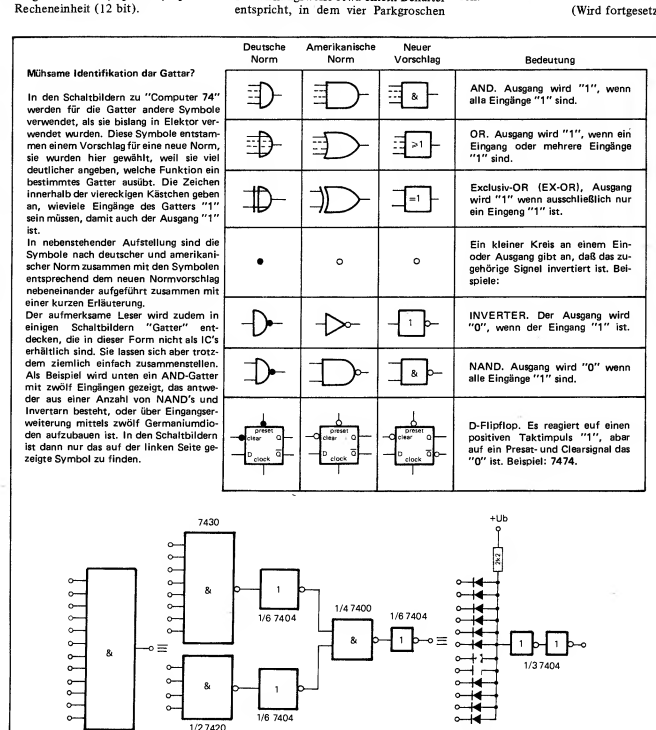
DST). Auch HALT ist wenigstens zum Teil eine periphere Einheit, weil es auf die programmierte Weise von SRC "angerufen" wird. Es greift aber direkt in den Ablauf des Mikroprogramms ein und steht somit "in der Nähe" der zentralen Steuerung.

Die nächsten Themen

In weiteren Abschnitten werden nachfolgende peripheren Einheiten behandelt:
Programm-Zähler, Speicher, 1 parallele Recheneinheit (12 bit).

Die Recheneinheit kann die folgenden Funktionen ausführen: Addition, Subtraktion, Multiplikation und Division, Inkrement und Dekrement; außerdem verschiedene logische Funktionen wie AND, OR, NAND, NOR, NOT, MASK, EX-OR usw. STACK und STACK pointer, womit sich auf einfache Weise ein Stack ("Push down list"; FILO = First in – Last out Speicher) programmieren läßt.
Ein STACK ist ein Stapelspeicher, dessen Wirkungsweise etwa einem Behälter entspricht, in dem vier Parkgroschen gestapelt werden. Wird ein DATA-Wort in einen solchen Speicher eingegeben, so schieben alle anderen Wörter um einen Platz weiter. Beim Auslesen ist nur das zuletzt eingegebene Wort erreichbar, worauf der Rest dann nachschiebt. Ein solches STACK weist große Vorteile bei verschiedenen Programmierschritten auf.
Indirektregister werden benötigt, um Adressen von SRC und DST nicht direkt, sondern indirekt zu programmieren.

(Wird fortgesetzt)



Mühsame Identifikation dar Gattar?

In den Schaltbildern zu "Computer 74" werden für die Gatter andere Symbole verwendet, als sie bislang in Elektor verwendet wurden. Diese Symbole entstammen einem Vorschlag für eine neue Norm, sie wurden hier gewählt, weil sie viel deutlicher angeben, welche Funktion ein bestimmtes Gatter ausübt. Die Zeichen innerhalb der viereckigen Kästchen geben an, wieviele Eingänge des Gatters "1" sein müssen, damit auch der Ausgang "1" ist.
In nebenstehender Aufstellung sind die Symbole nach deutscher und amerikanischer Norm zusammen mit den Symbolen entsprechend dem neuen Normvorschlag nebeneinander aufgeführt zusammen mit einer kurzen Erläuterung.
Der aufmerksame Leser wird zudem in einigen Schaltbildern "Gatter" entdecken, die in dieser Form nicht als IC's erhältlich sind. Sie lassen sich aber trotzdem ziemlich einfach zusammenstellen. Als Beispiel wird unten ein AND-Gatter mit zwölf Eingängen gezeigt, das antweder aus einer Anzahl von NAND's und Invertarn besteht, oder über Eingangserweiterung mittels zwölf Germaniumdioden aufzubauen ist. In den Schaltbildern ist dann nur das auf der linken Seite gezeigte Symbol zu finden.

Deutsche Norm	Amerikanische Norm	Neuer Vorschlag	Bedeutung
		&	AND. Ausgang wird "1", wenn alla Eingänge "1" sind.
		≥1	OR. Ausgang wird "1", wenn ein Eingang oder mehrere Eingänge "1" sind.
		=1	Exclusiv-OR (EX-OR), Ausgang wird "1" wenn ausschließlich nur ein Eingeng "1" ist.
●	○	○	Ein kleiner Kreis an einem Ein- oder Ausgang gibt an, daß das zugehörige Signel invertiert ist. Beispiele:
		1	INVERTER. Der Ausgang wird "0", wenn der Eingang "1" ist.
		&	NAND. Ausgang wird "0" wenn alle Eingänge "1" sind.
preset clear Q D clock Q̄	preset clear Q D clock Q̄	preset clear Q D clock Q̄	D-Flipflop. Es reagiert euf einen positiven Taktimpuls "1", abar auf ein Presat- und Clearsignal das "0" ist. Beispiel: 7474.

Teil 4

Computer 74

J.T.W. Damen

Die ersten drei Teile des Artikels behandelten das "Herz" des Computers, die zentrale Steuereinheit. Im vorliegenden und in den folgenden Teilen ist eine Anzahl "peripherer Einheiten" an der Reihe.
Zunächst werden Programmzähler und Speicher besprochen. Der Programmzähler ist ein Zähler, der die Programmschritte "anwählt". Für die Ausführung des Speichers bestehen zwei Möglichkeiten: Eine verhältnismäßig preiswerte Lösung mit dynamischen Schieberegistern, oder eine teurere - aber viel schnellere - mit Random Acces Memories (RAM's).

Der Programmzähler (PC)

Wie bereits in den vorangegangenen Abschnitten mehrfach erwähnt, weist der Programmzähler (PC) immer die nächstfolgende "Instruktion" an, wenn der Computer ein Programm abarbeitet. Es handelt sich hier um einen Zähler, der stets automatisch einen Schritt weiterzählt, wenn die zentrale Steuereinheit ein Wort aus dem Speicher gelesen hat. Als vollwertige periphere Einheit muß der PC die richtigen Antworten auf die Signale von der zentralen Steuereinheit geben. Das bedeutet im einzelnen:
Wird die Adresse des PC (= 7776) auf die SRC-Leitungen gesetzt und wird anschließend SRCENB 1, dann muß der PC das detektieren. Der Inhalt des Zählers gelangt auf die DATA-Leitungen anschließend: SRCRDY = 10. Dieser Zustand bleibt aufrecht erhalten bis der PC feststellt, daß SRCENB wieder 0 wird. Dann läßt der PC auch SRCRDY auch wieder auf 0 gehen und unmittelbar anschließend wird der Zähler um eins erhöht. Die Funktion des PC als SRC ist damit vollendet.
Der PC kann aber nicht nur als SRC angerufen werden, sondern auch als DST, somit läßt sich der Inhalt verändern. Auf diese Weise ist es möglich, einen Sprung im laufenden Programm auszuführen, Beispiel:

Speicher-Adresse	Inhalt	Name der Speicher-Adrasse	Inhalts-bezeichnung	Bemerkungen
.0012	....		...	Programm
0013	....		...	
0014	0016		S	(SRC)
0015	7776		PC	(DST)
0016	0200	S,	SPRUNG	Inhalt von S ist die Sprungadresse
....	....			
0200	....	SPRUNG	...	Programmsprung zur
0201	....		...	Speicher-Adresse 0015

Hat die zentrale Steuereinheit die aus SRC & DST bestehende "Instruktion" der Speicherplätze 0014 und 0015 gelesen, so steht der PC auf 0016. Wird anschließend die "Instruktion" ausgeführt, so wird SRC "S" mit DST "PC" verbunden, oder anders gesagt: der Inhalt des Speicherplatzes 0016 (=0200) wird in den PC eingelesen. Nach Ausführung dieser Instruktion steht der PC also auf 0200. Die nachfolgende "Instruktion" wird auf 0200 & 0201 aus dem Speicher ausgelesen.
Selbstverständlich wird der PC als DST nicht um eins erhöht. Die Arbeitsweise des PC in Kurzform:

als SRC
- warten bis SRC :=7776; (:= bedeutet "wird"),
- warten bis SRCENB := 1;
- Zählerinhalt auf DATA-Leitungen setzen;
- mache SRCRDY = 1;
- warten bis SRCENB := 0;
- mache SRCRDY = 0 (und mache die DATA-Leitungen wieder frei);
- erhöhe den Zähler um eins.

und als DST:
- warte bis DST := 7776;
- warte bis DSTENB := 1;
- setze die auf den DATA-Leitungen vorhandene Information in den Zähler;
- mache DSTRDY = 1;
- warte bis DSTENB := 0;
- mache DSTRDY = 0.

Auch im Programm kann der PC als DST angerufen werden (das ist nicht das Privileg der zentralen Steuereinheit!). Dann ist aber zu beachten, daß der PC nach Ausführung der Instruktion einen Speicherplatz überschlage hat. Denn: Die zentrale Steuereinheit liest zweimal den PC, dann weist dieser das nächste Speicherwort an. Erst dann wird die Instruktion ausgeführt, zufolge der der Speicher wieder gelesen und um ein Wort erhöht wird. Beispiel:

Speicher-Adresse	Inhalt	Name der Speicher-Adresse	Inhalts-bazeichnung	Bamerkungan
....	....			
0012	7776		PC	(SRC)
0013	0017		HELP	(DST)
0014				wird überschlagen
0015	....		...	(SRC)
0016	....		...	(DST)
0017	0000	HELP,	0	

Nach Ausführung der Instruktion auf den Speicherplätzen 0010 & 0011 ist der Inhalt des PC 0012. Der Computer liest die folgende Instruktion (auf 0012 & 0013), wonach der PC auf 0014 steht. Dann wird die Instruktion ausgeführt: Der Inhalt des PC (=0014) wird auf Speicherplatz 0017 ("Hilfe") transportiert. Das Resultat:
Inhalt PC = 0015
Inhalt "Hilfe" (0017) = 0014.
Speicherplatz 0014 wird überschlagen, er kann für andere Zwecke benutzt werden.
Dieser Ablauf kann bei Subroutine-Sprüngen nützlich sein, die vielleicht später an die Reihe kommen sollen (Subroutine ist ein Programmteil, losgelöst vom Rest, das an jedem beliebigen Platz innerhalb des Programms "zwischenzeitlich" abgewickelt werden kann). Auch die unmittelbar auf eine Instruktion folgende Eingabe einer Konstanten oder einer Zahl kann damit ausgeführt werden (sog. unmittelbare Instruktion; immediate instruction).

Die Schaltung

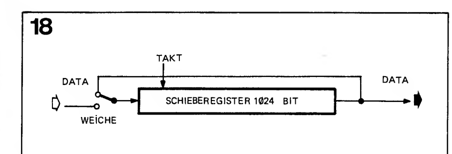
Wie jede periphere Einheit ist auch der PC mit einer ADRESSENAUSWAHL-Schaltung versehen, einer für SRC und einer für DST. Diese Schaltung wurde bereits beim Switch-Register (siehe dazu Bilder 11 und 12) besprochen. Die Adressen-Auswahl detektiert, daß der PC angerufen wird. Das Herz des PC wird vom Zähler gebildet, er besteht aus drei SN 74161 (Bild 17). Diese IC's zählen bei Steuerung über den Takteingang CK (Clock) in üblicher Weise, es kann aber auch über Parallel-Eingänge (A, B, C, D) willkürlich auf einen Zählerstand gesetzt werden (Load-Eingang L). Vergleich hierzu: der Adressenzähler bei dem ROM.
Steht die Adresse 7776 (PC) auf den SRC-Leitungen und wird SRCENB 1, dann werden über DEVice SELect die zwölf Gatter (SN 7438) geöffnet, damit gelangt der Zählerinhalt auf die DATA-Leitungen. SRCRDY wird durch das Funktionsflipflop in DEVSEL geliefert. Dieses Flipflop wird zurückgesetzt, wenn SRCENB "abfällt", daraufhin werden die Ausgangsgatter wieder gesperrt. Die negative Flanke am Ausgang des Funktionsflipflops wird in einen negativen Impuls umgesetzt (siehe Bild 14), dieser wird dem Takteingang des Zählers zugeführt. Der Zählerinhalt wird damit um eins erhöht und somit ist seine Aufgabe erfüllt.
Die Adresse von PC auf DST wird durch die DST-Adressenauswahl erkannt und wieder über eine DEVSEL-Schaltung in ein "Load"-Signal übersetzt. Die auf den DATA-Leitungen vorhandene Information wird in den Zähler übernommen. DSTRDY wird aktiviert und nachdem DSTENB 0 geworden ist, wird auch DSTRDY wieder 0. Es erscheint nun kein Taktsignal am CK-Eingang.

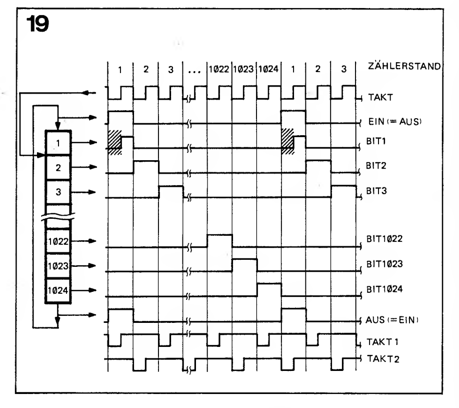
Der Speicher des Computers

Ein Computer ohne Speicher ist wie ein Amateurkoch der kein Kochbuch besitzt. Der Koch benötigt die Rezepte um seine Künste zu einem guten Ende

17

Bild 17. Schematische Darstellung des Programmzählers (PC). Dabei handelt es sich um einen Zähler, der den Platz im Programm angibt. Wird er als SRC angewiesen, so wird der Zählerinhalt auf die DATA-Leitungen gesetzt; sobald SRCENB abfällt, wird der der Zählerinhalt automatisch um eins erhöht. Wird er als DST angewiesen, so werden die Informationen auf den DATA-Leitungen als neuer Zählerinhalt übernommen, weil ein "LOAD"-Befehl gegeben wird. Auf diese Weise sind Sprünge innerhalb des Programms möglich.





zu führen, der Computer findet die auszuführenden Arbeitsgänge im Speicher in Form von Instruktionen aufgelistet. Der Amateurkoch kann notfalls die Hilfe seiner Frau in Anspruch nehmen, um den nächsten "Programmschritt" zu erfahren. Ein Computer könnte beispielsweise einen Cassettenrekorder einschalten, um die auf dem Band gespeicherten Instruktionen abzufragen. Legt man aber Wert auf eine flotte Arbeitsweise, so ist für den Computer ein Speicher unentbehrlich!
Als Speicher für den Computer 74 wurden integrierte MOS-Schaltkreise gewählt. Zwar bestehen auch andere Möglichkeiten, sie erfordern aber zumeist einen solchen Aufwand, der die Angelegenheit stark verteuert. Außerdem spielt die Beschaffungsfrage eine nicht unwesentliche Rolle. Damit soll aber keinesfalls gesagt werden, daß fortschrittliche Speichersysteme nicht verwendbar sind!
Hier werden zwei Speichersysteme behandelt, die erste - preiswertere Version - wird ausführlich beschrieben. Die zweite - teurere Version - läßt sich viel einfacher aufbauen.
Bei der ersten Ausführung handelt es sich um einen Schieberegister-Speicher, die zweite ist mit Random-Access-Memories (RAM's) aufgebaut, verwendet werden Intel-IC's Typ 2102. (RAM bedeutet: Speicher mit wahlfreiem Zugriff).

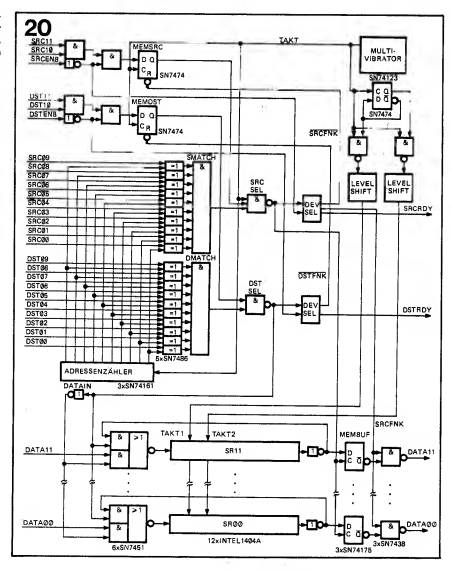
Schieberegister

Dynamische 1024 bit-Schieberegister werden von verschiedenen Firmen angeboten (u.a. von Intel und National Semiconductors), relativ preiswert ist der Typ 1404 A. Mit zwölf dieser MOS-IC's läßt sich ein Speicher von 1 k Wörtern zu 12 bit aufbauen (k bedeutet bei Computer-Speichern: 1024).
Die Arbeitsweise eines Schieberegisters ist ziemlich einfach zu erklären: Der Eingang ist über eine "Weiche" mit dem Ausgang verbunden. Ist das Schieberegister als Ringzähler geschaltet, dann kreist die eingespeicherte Information im Register. Alle bits passieren den Ausgang in serieller Folge, sie sind dort auslesbar. Neue Informationen können über die Weiche am Eingang eingegeben werden (Bild 18). Soll ein bit geändert werden, so wird abgewartet, bis es am Ausgang erscheint. Bevor es wieder in den Kreislauf gelangt, wird die Weiche umgelegt und die neue Information tritt an die Stelle der alten Information. Wird die Weiche nun wieder umgelegt, so ist die neue Information eingespeichert. Auf diese Weise entsteht ein Speicher für 1024 Wörter zu 1 bit. Werden zwölf solcher Schaltungen gleichzeitig gesteuert, so sind stets 12 bit gleichzeitig verfügbar. Es ergibt sich somit ein Speicher für 1024 Wörter zu 12 bit. Mit n Schieberegistern läßt sich also ein Speicher für 1024 Wörter zu n bit aufbauen. Wie ist ein solcher Speicher nun zu adressieren?
Zuvor aber etwas prinzipielles über ein Schieberegister. Das Register wird durch ein Taktsignal gesteuert, in Wirklichkeit sind es zwei Taktsignale, aber darüber später mehr. Jedesmal wenn der Taktimpuls von 1 auf 0 geht (negative Flanke), werden alle 1024 bit um einen Platz weitergeschoben, somit erscheint auch bei jedem Taktimpuls ein neues Wort am Ausgang, d.h., bei einem einzigen Schieberegister ist das ein "Wort" zu 1 bit. Beim nächsten Taktimpuls, und zwar beim Übergang von 0 nach 1 (positive Flanke) wird die am Registereingang stehende Information in das Register "eingetaktet". Anschließend wiederholt sich das Spiel. Dieser Zyklus ist in Bild 19 in Form eines Impulsdiagramms dargestellt, dabei bleibt der "Zählerstand" außerhalb der Betrachtung. Der Taktimpuls besteht in Wirklichkeit aus zwei sich überlappenden Impulsfolgen, Takt 1 und Takt 2. Im "Innenleben" des 1404 A wirken sich die beiden Impulsfolgen so aus, wie der in Bild 19 (oben) gezeichnete Taktimpuls.
Das 1404 A ist ein dynamisches Schieberegister, das heißt: die Information wird als Ladung von internen Kapazitäten gespeichert. Diese Ladung muß innerhalb einer ganz bestimmten Zeitdauer "nachgefüllt" werden. Bleiben die Taktimpulse länger als etwa 10 ms aus, so leckt soviel Ladung aus den Kapazitäten heraus, daß die Information verlorengeht. Der Taktgenerator darf daher nicht gestoppt werden, die Informationen müssen ständig "kreisen".
Zum Adressieren ist ein Zähler erforderlich, der die Taktimpulse für das Schieberegister zählt. Der Zähler wird zu einem bestimmten Zeitpunkt auf 0 gesetzt, und von diesem Augenblick an darf der Zählvorgang nicht mehr unterbrochen werden. Das bit oder das Wort, das in diesem Moment am Ausgang

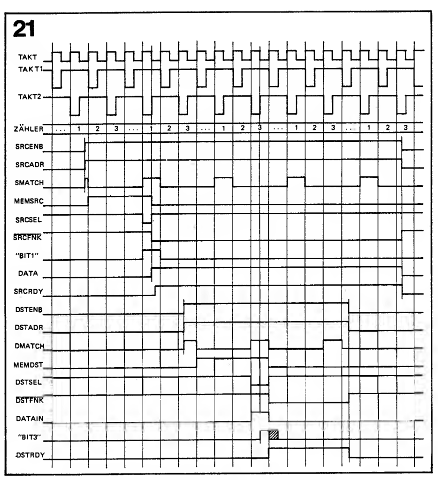
Bild 18. Das Prinzip eines Schieberegister-Speichers. Bei jedem Taktimpuls wird der Speicherinhalt eine Stelle weitergeschoben, so daß der nächste "Speicherplatz" am Ausgang erscheint.

Bild 19. Das Impulsdiagramm eines rückgekoppelten Schieberegisters.

Bild 20. Das vollständige Prinzipschaltbild eines 1 k X 12 bit Speichers. Links oben die Adressenauswahl für den Speicher, in der Mitte die Adressenauswahl für den Speicherplatz, rechts oben die Steuerung und unten der Speicher mit den Ein- und Ausgangsbuffern.



vorhanden ist, erhält die Adresse 0, das nächstfolgende die Adresse 1, usw. Der Zähler muß bis 1024 zählen, es handelt sich also um einen 10 bit-Zähler.
Das Adressieren des Speichers geht so vor sich: Die angebotene Adresse desjenigen Wortes, das ausgelesen oder verändert werden soll, wird mit dem Zählerstand verglichen. Sowie Übereinstimmung herrscht, ist das betreffende Wort am Ausgang verfügbar. Soll das Wort gelesen werden, so wird es in einen Ausgangsspeicher eingegeben (D-Flipflops), von dort aus kann es auf die DATA-Leitungen gesetzt werden. Soll das Wort geändert werden (bei einem Speicher sagt man besser: soll für diese Adresse eine neue Information eingegeben werden), wird die Weiche am Eingang umgelegt, bevor die alte Information wieder in das Schieberegister eingetaktet wird. Anstatt dieser wird die neue Information eingegeben und anschließend sofort wieder die Weiche umgelegt, so daß das nächstfolgende Wort nicht verlorengehen kann. Eine der Schwierigkeiten bei solchen Speichern besteht in der ziemlich kleinen Zeitspanne, während der die Weichen umgelegt werden können und während der die neue Information bereitstehen muß. Diese Zeitspanne ist in Bild 19 durch Schraffur gekennzeichnet. Die Weichen werden mit TTL-IC's aufgebaut (AND-OR-INVERTER SN 7450 oder SN 7451), sie sind beträchtlich schneller als die relativ langsamen MOS-Schieberegister. Die angegebene Zeitspanne entspricht der Dauer eines Taktimpulses, als Mindestforderung gelten hier 170 ns. Während dieser Zeit kann TTL eine ganze Menge Dinge tun.
Ein etwas schwierigerer Punkt ist die Synchronisation, die Schwierigkeit besteht darin, daß die Adresse (von außen her) gerade dann, und exakt zu dem Zeitpunkt angeboten wird, wenn der Zähler den zugehörigen Stand erreicht hat. Dieser Speicherplatz muß unmittelbar ausgelesen oder beschickt werden. Da das mit der Flanke des Taktimpulses geschieht, ist keine ausreichende Sicherheit gegeben, daß das Auslesen oder das Eintakten zum richtigen Zeitpunkt erfolgt. Dafür ist aber Sorge zu tragen, eine Adresse muß mindestens während der gesamten Dauer des Taktimpulses vorhanden sein (Taktimpulsdauer = Zählerstand), bevor zum Lesen oder Schreiben übergegangen wird. Die vollständige Schaltung (Bild 20) zeigt, wie sich der Vorgang in der Praxis abspielt.
Es sind zwei Eingangs-AND's vorhanden, eins für SRC und eins für DST. Sie stellen die Eingänge für die zwei "most significant bits" der Adresse dar: für SRC 11 und SRC 10, bzw. für DST 11 und DST 10. Unter "most significant bits" sind diejenigen bits zu verstehen, die am "weitesten links" stehen. Die Adressen für den gesamten Speicher laufen von 0000 bis 1777 (Binär: 000 000 000 000 bis 001 111 111 111). Eine Adresse für den Speicher ist daher vorhanden, wenn die beiden genannten bits 0 sind, das AND-Gatter detektiert es. Die SRC-Signale kommen direkt von BUS, die BUS-Signale sind invertiert. Die übrigen 10 bit der Adresse gelangen



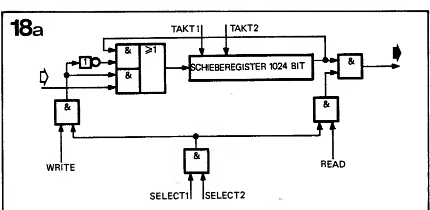


Bild 21. Das Impulsdiagramm für den Speicher nach Bild 20. Es wird zuerst Speicherplatz 1 ausgelesen, im Diagramm erfolgt das bei dem zweiten SMATCH-Impuls. Die Information wird dann auf die DATA-Leitungen gesetzt, anschließend wird Speicherplatz 3 als DST angewiesen. Beim zweiten DMATCH-Impuls werden die DATA in Speicherplatz 3 eingelesen.

Bild 18 a. 1014 bit-Schieberegister mit integrierter "recirculating"-Logik (z.B. IM 7712). Siehe dazu auch Bild 18.

Bild 20 a. 1 k X 12 bit Speicher, aufgebaut mit Schieberegistern des Typs IM 7712. (Siehe dazu auch Bild 20).

Bild 21 a. Impulsdiagramm für einen Speicher mit IM 7712 (Siehe auch Bild 21).

an Vergleicher-Schaltungen die mit SN 7486 Exklusiv-OR-Gattern aufgebaut sind. Es sind 10 für SRC und 10 für DST.
Die anderen Eingänge der EX-OR's sind mit den Ausgängen eines 10 bit-Zählers, dem Adressenzähler, verbunden. Die Ausgänge der EX-OR's steuern ein AND-Gatter mit 10 Eingängen (NAND mit 8 Eingängen + NAND mit 2 Eingängen + Inverter). Ist der Ausgang des Gatters I so herrscht Gleichstand zwischen Adresse und Zählerstand (MATCH). Dieses Signal muß aber noch das Gatter SRCSEL passieren, das durch den Ausgang eines D-Flipflops gesperrt wird. Dieses Flipflop sorgt für die erforderliche Synchronisierung. Das AND-Signal von SRC 11, SRC 10 und SRCENB wird dem D-Eingang zugeführt, mit der positiven Flanke des Taktimpulses wird diese Information in das Flipflop eingetaktet und erscheint an seinem Ausgang. Die gleiche Taktflanke erhöht den Zählerstand, so daß der Zählerstand im Gleichschritt ist. Das Impulsdiagramm Bild 21 verdeutlicht die Zusammenhänge. Hier ist zu erkennen, was geschieht, wenn eine Adresse angeboten wird und der Zählerstand den gleichen Wert aufweist; diese Konstellation bleibt unbeachtet, ein vollständiger Speicherzyklus muß abgewartet werden (1024 Taktimpulse). Erst dann wird das erneut erscheinende MATCH-Signal SRCSEL passieren können (Memory source select). Dem NAND SRCSEL wird noch ein drittes Signal, der Taktimpuls, zugeführt, damit erhält das Ausgangssignal die halbe Länge eines Zählerstandes. Damit entsteht ein positiver Impuls, und zwar mitten innerhalb der Zeitspanne, innerhalb derer das gesuchte Wort am Speicherausgang steht (Bild 21). Diese Flanke dient als Taktimpuls für die 12 D-Flipflops, in denen das zu lesende Speicherwort gepuffert ist. Die gleiche positive Flanke von SRCSEL schaltet einen DEVice-SELector ein. Der komplementäre Ausgang des Zustandsflipflops im DEV-SEL liefert das Reset-Signal für MEMSRC, so daß garantiert nur jedesmal ein Wort aus dem Speicher ausgelesen wird. Diese Leitung bleibt nämlich 0, bis der gesamte SRC-DST-Zyklus abgelaufen ist. SRCRDY wird wieder von DEVSEL geliefert, und das "Frage-und-Antwort-Spiel" zwischen SRCENB und SRCRDY wird abgearbeitet.
Auf der DST-Seite verlaufen die Dinge praktisch identisch (Speicher als DST heißt: ein neues Speicherwort schreiben). Auch hier ist ein MEMDST-Flipflop vorhanden, dessen Ausgangssignal von MATCH öffnet. DSTSEL ist nicht mit Takt verbunden: ein Impuls mit der Dauer einer vollen Taktperiode wird benötigt, um die Weichen umzulegen (SN 7450 oder 7451). Für die Dauer des Zählerstandes ist die Information auf den DATA-Leitungen mit den Schieberegister-Eingängen verbunden. Der Taktimpuls selbst bewirkt nun, daß genau inmitten dieser Zeitspanne das neue Speicherwort eingetaktet wird (Positive Flanke!).

Die Polarität der Daten vor und hinter dem Speicher ist etwas verwirrend, auf den DATA-Leitungen (BUS) ist die Information negativ (Negative Logik: 1 = 0 Volt). Sie erscheint positiv an den Ausgängen der AND-OR-INVERT-Gatter. So werden die Wörter auch in den Speicher eingelesen; die gespeicherte Information ist positiv (Positive Logik: "1" = 3 V ... 5 V; TTL-Pegel). Wiederum invertiert gelangen sie zurück an den anderen Eingang des AND-OR-INVERT-Gatters. Negativ wird die Information auch in den MEMory BUFfern gespeichert. Die Daten gelangen als positive Signale von den Komplementär-Ausgängen der Buffer an die Treiber-Gatter SN 7438. Schließlich wird alles, letztmalig invertiert, als DATA auf BUS gesteuert.

In Bild 21 ist ein Beispiel angegeben für die Instruktion:

(SRC) – Speicher 1
(DST) – Speicher 3.

Hier sind alle "worst-case"-Situationen vorgesehen (die Wirklichkeit sieht ganz anders aus!):

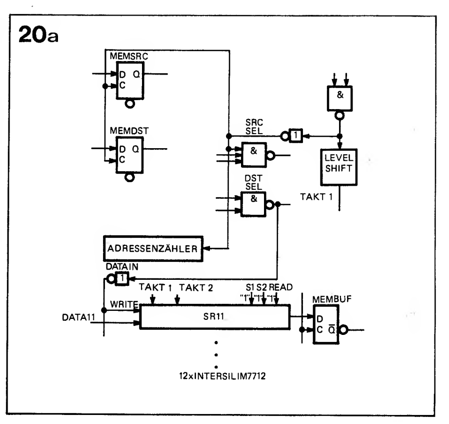
1 – SRC und SRCENB der Adresse 1 werden während des Zählerstandes 1 angeboten, und zwar in der zweiten Hälfte;

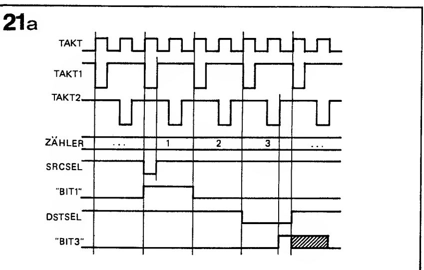
2 – DST und DSTENB der Adresse 3 werden während des Zählerstandes 3 angeboten, und zwar während der ersten Hälfte;

3 – DSTENB bleibt länger als ein vollständiger Speicherzyklus stehen. Es gibt zwar noch ein MATCH, das sich aber nicht auswirken kann.

4 – SCRENB bleibt länger als ein vollständiger Speicherzyklus stehen. Es gibt dann mehr als einen SMATCH-Impuls, von denen allerdings kein einziger beantwortet wird.

"Bit 1" und "Bit 3" bedeutet: Sind diese "Signale" 1, so ist die zugehörige Information (Speicherwort 1 und 3) an Ein- und Ausgang des Speichers vorhanden. In Bild 20 sind zwei Blöcke mit "LEVEL SHIFT" bezeichnet, es handelt sich um Schaltungen zur Anpassung der –9 V-Taktsignale an die Schieberegister 1404 A. Vergleicht man Bild 19 mit Bild 21, so wird man feststellen, daß die mit "TAKT" bezeichneten Signale sich nicht gleichen, sie sind komplementär zueinander. Das Taktsignal zum Schieben (und Eintakten) für den Speicher ist das addierte doppelte Taktsignal: TAKT 1 und TAKT 2. "TAKT" in Bild 21 geht zu beiden MEM-D-Flipflops, zu SCRSEL und zum Adressenzähler. Mit Hilfe einer Teiler-Schaltung werden TAKT; und TAKT 2 daraus abgeleitet. Die 1404's arbeiten mit den Speisespannungen +5 V und –9 V. Anstatt des Typs 1404 von Intel (oder äquivalenter





Typen anderer Hersteller) lassen sich auch Schieberegister mit anderer Ausführungsform verwenden: Schieberegister mit eingebauter "recirculating"-Logik. Die Weiche (Bild 18) und der Rückführungspfad sind in den Baustein integriert. In diesem Fall können die AND-OR-INVERT-Gatter entfallen, die in Bild 20 die Weichen bilden.

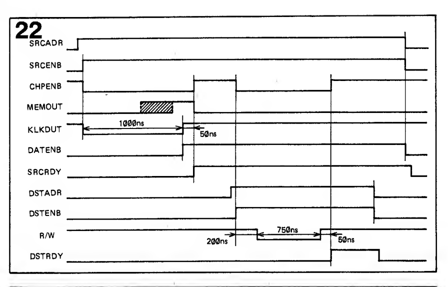
Als Beispiel für diesen Typ kann das IM 7712 von Intersil dienen (Bild 18 a). Im 1404 werden die beiden Taktimpulsfolgen innerhalb des IC's ge"multiplext" (sie können daher als insgesamt ein Takt betrachtet werden). Das ist bei dem IM 7712 nicht der Fall; hier sind für das Durchschieben eines bits zwei Taktimpulse erforderlich, an jedem Takteingang einer.

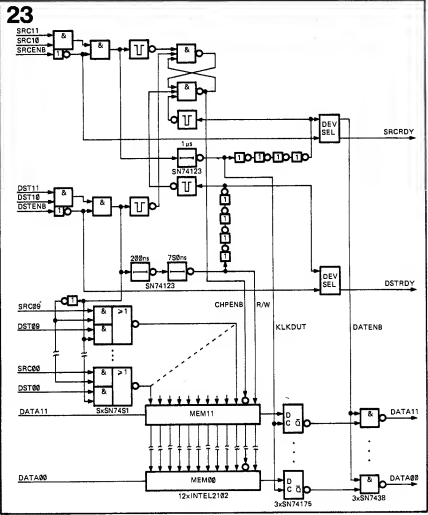
TAKT 1 tritt hier als Ausgangstakt auf; mit seiner negativen Flanke erscheint der Inhalt der Speicherzelle Nr. 1024 am Ausgang. Als Eingangs-Takt dient TAKT 2: Hier wird mit der positiven Flanke die Eingangsinformation in das Schieberegister übernommen. Außerdem schieben dann alle Zellen ihre Information einen Schritt weiter.

Gegenüber Bild 20 gilt für IM 7712 die Schaltung nach Bild 20 a:

– die AND-OR-INVERT-Weichen entfallen, ebensowie der Rückführungspfad.

– TAKT für MEMSRC, MEMDST,





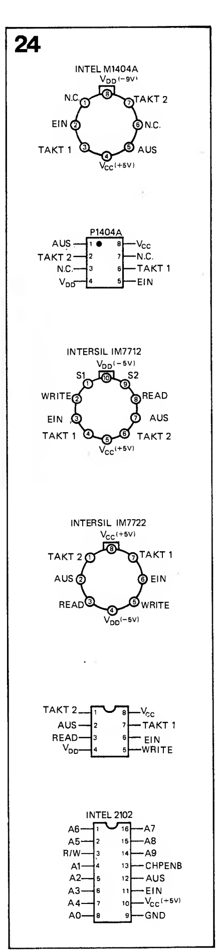


Bild 22. Impulsdiagramm eines 2102-Speichers (statisches RAM; Random Access Memory).

Bild 23. Das vollständige Prinzipschaltbild aines mit RAM's (Typ 2102) aufgebauten Speichers.

Bild 24. Anschlußbalegung verschiedener Speichar-IC's.

SCRSEL und Adressenzähler werden von TAKT 1 abgeleitet: alles wird mit dem Ausgangstakt synchronisiert. Ansonsten ist die Arbeitsweise mit der des 1404 vergleichbar, mit dem Unterschied, daß bei einer negativen Flanke von TAKT 1 die Information am Ausgang erscheint, und daß mit der positiven Flanke von TAKT 2 die Informa-

tion eingelesen wird (Siehe auch Bild 21 a).
Ferner sei angemerkt, daß es die "INVERT"-Funktion der AND-OR-INVERT-Weichen im 7712 nicht gibt, dadurch entfallen auch die Inverter hinter jedem Schieberegister. Die Polarität ist nun folgendermaßen:
auf den DATA-Leitungen: negativ,
in den Schieberegistern: negativ(!),
im MEMory BUFfer: negativ,
zu den 7483-Gattern: positiv,
zu den DATA-Leitungen: negativ.
Der Typ IM 7722 (auch Intersil) ist dem IM 7712 nahezu gleichwertig, es fehlen aber die beiden Select-Eingänge (Bild 18), anstatt 10 Anschlußleitungen sind nur 8 vorhanden. Von Signetics ist der Typ 2512 identisch mit dem IM 7712 und der Typ 2525 mit dem IM 7722. Auch das 1405 A von Intel ist ein "recirculating"-Schieberegister, aber mit einer Länge von 512 bit.
Ferner sind die Typen 1402 A und 1403 A mit dem 1404 äquivalent, alle 1024 bit, nur anders organisiert: 1402 A 256 X 4 bit, (4 Parallel-Ein- und Ausgänge) 1403 A: 512 X 2 bit (2 Parallel-Ein- und Ausgänge). Werden Eingänge sowie Ausgänge jeweils untereinander verbunden, so können die letztgenannten Typen als Ersatz für 1404 A dienen. Die Anschlußbelegung der verschiedenen Typen ist aus Bild 24 zu entnehmen.

Random-Access-Memories

Hier folgt nun die Kurzbeschreibung eines Speichers, der mit 12 Stück RAM's des Typs 2102 von Intel aufgebaut ist. Dieses IC ist ein Speicher mit wahlfreiem Zugriff, er ist TTL-kompatibel an allen Eingängen und an allen Speisespannungsanschlüssen. Er hat 10 Adressen-Eingänge, einen CHiP-ENaBle-Eingang und einen Lese/Schreib-Eingang (Read/Write), das sind zusammen mit den beiden Speisespannungsanschlüssen 14 pins.
Das Impulsdiagramm (Bild 22) zeigt die zeitlichen Verhältnisse, wie sie für eine gute Arbeitsweise gegeben sein müssen. Die Anforderungen in Zusammenfassung:

Als Konsequenz aus diesen Anforderungen ergeben sich die in Bild 22 angegebenen Zeiten. Eine Möglichkeit für die schaltungstechnische Realisation ist in Bild 23 angegeben. Die Zeiten werden
– durch Monoflops erzeugt (1000, 200, 750 ns) oder
– durch Verzögerung mit 4 Invertern (etwa 50 ns). Die Blöcke, in die ein negativer Impuls eingezeichnet ist, stellen Flankendetektoren dar, wie sie bereits besprochen wurden (Bild 14).
Bei dem Typ 2102 handelt es sich um statische Speicher, das heißt: Die gelesenen Daten bleiben noch solange an den Ausgängen stehen, wie Adresse, Chip-enable und Read/Write angeboten sind. Daher ist es möglich, den Speicherinhalt ohne Zwischenspeicher (Buffer) direkt auf die Data-Leitungen zu setzen (über Treiber SN 7438). Das führt aber zu Beschränkungen bei der Benutzung des Speichers, denn es ist dann möglich, innerhalb einer Instruktion (SRC & DST) Daten von einem Speicherwort zu einem anderen zu übertragen. Es müßten dieselben Adresseneingänge gleichzeitig benutzt werden, und das geht selbstverständlich nicht. Deshalb ist in Bild 23 wieder ein Buffer vorhanden, damit ergeben sich die gleichen Möglichkeiten wie bei einem Schieberegisterspeicher.
Die Vorteile des RAM's liegen vor allem im einfacheren Aufbau, und besonders im wahlfreien Zugriff: keine kniffligen Synchronisier-Schaltungen, außerdem ist ein solcher Speicher viel schneller (max. 1000 ns Zugriffszeit gegenüber 300 μs und gemittelt 150 μs bei dem Schieberegister, die maximale Taktfrequenz beträgt 3 MHz). Zudem bedeutet es eine große Erleichterung, daß alle Ein- und Ausgänge TTL-kompatibel sind.
Der einzige Nachteil liegt im wesentlich höheren Preis, er schlägt natürlich bei 12 Stück zu Buch! Es ist aber festzustellen, daß sich die Preise nach unten bewegen.

(Wird fortgesetzt).

	MIN	TYP	MAX	
– ADRES & CHPENB: = 0 bis DATA OUT		500	1000 ns	(access-time)
– CHPENB: = 1 bis DATA OUT	00		ns	(previous data valid)
– ADRES bis R/W: = 0	200		ns	(write setup time)
– R/W = 0	750		ns	(write pulse width)
– R/W: = 1 bis CHPENB: = 1	50		ns	(write recovery time)
– DATA IN bis R/W: = 1	800		ns	(data setup time)

NACH-LESE

Erfahrungen mit, Berichtigungen von und Nachträge zu Elektor-Publikationen

Bildschirm-Tennis

Beim Abdruck der Aktionsschaltung Nr. 172 (Heft 10/74, Seite 10–50) sind einige Bauteilwerte irrtümlich nicht angegeben. Die fehlenden C-Werte sind in den nachfolgenden Tabellen zusammengefaßt.

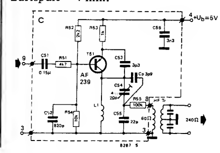
Tabelle zu Bild 1.

	Horizontal-Generator	Vertikal-Generator
C_{11}	15 n	4,7 μ
C_{12}	15 n	4,7 μ
C_{13}	360 p	22 n

Tabelle zu Bild 2.

	Video-Generator Horizontal		Video-Generator Vertikal	
	C_{21}	C_{22}	C_{21}	C_{22}
Ball (D, E)	1,4 n	160 p	0,47 μ	68 n
rechter Spieler (F, G)	1,4 n	180 p	0,47 μ	0,22 μ
linker Spieler (H, I)	1,4 n	180 p	0,47 μ	0,22 μ

Der in Bild 4 fehlende Widerstandswert für R_4 beträgt 56 k. Die Spule L_1 (Bild 5) besteht aus 1 . . . 2 Windungen; gegebenenfalls muß mit der Spule etwas experimentiert werden.
Anm. der Redaktion: Durch geringe Änderungen der frequenzbestimmenden Bauelemente des VHF-Oszillators arbeitet dieser im Fernsehband I (48,25 . . . 62,25 MHz).
Der Wert des Kondensators C_{53} ändert sich dann auf 3,3 p, während für C_{55} 22 p eingesetzt werden. Ein Trimmer (4 . . . 20 p) mit einem Parallelkondensator C_p = 3p9 ersetzen den Kondensator C_{54}. Mit diesen Änderungen muß die Induktivität der Spule L_1 bei 1 μH liegen. Wickeldaten der Spule: 20 Wdgn. 0,2 mm ϕ CuL, Wickeldurchmesser der Luftspule = 4 mm.



NACH-LESE

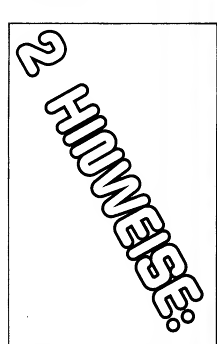
Erfahrungen mit, Berichtigungen von und Nachträge zu Elektor-Publikationen

Abschaltautomat für das Fernsehgerät

Die Aktionsschaltung Nr. 173 (Oktober '74, Seite 10–52) ist mit einer falschen Autorenangabe versehen. Autor der Schaltung ist Herr G. Osswald, Stuttgart.

Der Pawlowsche Hund, Heft 46

In der Texterläuterung zur Schaltung 1, Seite 10-35, sind die Indices der Transistoren falsch angegeben. Richtig muß es dort heißen: T_4 statt T_1, T_5 statt T_2, T_7 statt T_3, T_6 statt T_4, T_8 statt T_5, T_9 statt T_6.



- Am 24. und 31. Dezember 1974 entfällt die redaktionelle Fragestunde.
- Auf die Innenseite der Banderole, die als Verpackung und zum Adressieren des vorliegenden Heftes diente, ist eine "Wegwerfschaltung" gedruckt, wie in Heft 47, Seite 11–25 angekündigt. Achten Sie auch in Zukunft darauf, ob die Banderole eine evtl. nützliche Information für Sie enthält.

J.T.W. Damen

TEIL 5

COMPUTER 74

In dieser Folge wird die Arbeitsweise der Recheneinheit beschrieben. Die Recheneinheit hat die Aufgabe, logische Verknüpfungen und algebraische Operationen durchzuführen: u. a. AND, NAND, OR, NOR, EXCLUSIVE OR, Inkrement (1 addieren), Dekrement (1 subtrahieren), Addition und Subtraktion. Die Recheneinheit wurde für die doppelte Zeichenlänge von 24 Bit ausgelegt. Ferner können Zahlen mit einer Länge von 12 Bit multipliziert werden, das Ergebnis hat maximal 24 Bit; Zahlen mit einer Länge von maximal 24 Bit können durch 12 Bit-Zahlen dividiert werden, wobei das Ergebnis 24 Bit und der Rest 12 Bit lang sein können.
In der nächsten Folge wird dann die praktische Schaltungsausführung behandelt werden.

Alle logischen und algebraischen Operationen werden mit Hilfe der "arithmetic and logic unit" vom Typ SN 74 181 durchgeführt, nachstehend kurz ALU genannt. Dieses IC besitzt zweimal 4 Data-Eingänge (AØ . . . A3 und BØ . . . B3), 4 Funktionsausgänge (FØ . . . F3), 5 Steuereingänge (M, SØ . . . S3: ein "Modus"- und 4 "Select"-Eingänge), je einen "Carry"-Aus- und Eingang sowie einen A=B-Ausgang. Die Carry-Ein- und Ausgänge ermöglichen die Serienschaltung mehrerer ALU's und damit die Verarbeitung von Zahlen mit mehr als 4 Bit (Bild 25). Über die 4 S-Eingänge, den M- und den Carry-Eingang können insgesamt 48 verschiedene Funktionen gewählt werden. Sind z.B. M = "0", Carry = "0" und SØ . . . S3 = "0", "1", "1", "0", so wurde die Funktion "Addieren" gewählt. An den Ausgängen FØ . . . F3 (F) erscheint dann die Summe von AØ . . . A3 (A) und BØ . . . B3 (B) : F = A+B.
Läßt man den Carry-Eingang außer Betracht, so teilt M die Anzahl der möglichen Funktionen in zwei Gruppen. Mit M = "1" stehen die logischen (Boole'schen) Funktionen, bei M = "0" die arithmetischen Funktionen zur Verfügung. Jede Gruppe umfaßt 16 Funktionen, die über SØ . . . S3 gewählt werden können; insgesamt sind damit 32 Funktionen vorhanden. Benutzt man gleichzeitig den Carry-Eingang, so verdoppelt sich diese Zahl. Der Eingang wird allerdings nur bei den arithmetischen, nicht bei den logischen Funktionen benutzt. Das geschieht nicht willkürlich, denn das Carry-Signal stellt nichts anderes als den "1"-Übertrag bei den arithmetischen Operationen dar. Die Anzahl der Funktionen beträgt damit 48.
Bei mehreren ALU's in Serie wird der Carry-Ausgang des einen ALU mit dem Carry-Eingang des folgenden verbunden. Nur das Signal am Carry-Eingang des ersten ALU ist frei wählbar.

Schließlich besitzt der ALU noch einen "Ist Gleich"-Ausgang, der auf "1" liegt, wenn A und B gleich sind (AØ = BØ, A1 = B1 usw.). Dieser Ausgang ist von den Funktionsausgängen unabhängig und wird nur bei der Funktion "Subtrahieren" verwendet.
Bei den logischen Funktionen (M = "1") wird das Ergebnis "bitweise" ermittelt, d.h. für jedes Bit getrennt. Gesucht sei z.B. die UND-Funktion (M = "1", SØ = "0", S1 = "0", S2 = "1", S3 = "0", carry = "0" oder "1") von A = 0111 und B = 1100 (AØ = "1", A1 = "1", A2 = "1", A3 = "0" und BØ = "0", B1 = "0", B2 = "1", B3 = "1"), dann ist F = 0100 : FØ = AØ · BØ (logisches Produkt) = 1 · 0 = 0, F1 = A1 · B1 = 1 · 0 = 0 usw. Dagegen werden A und B bei den arithmetischen Operationen als Zahlen von 4 Bit interpretiert, so daß dann A und B wie in folgendem Beispiel addiert werden:

```
  0111
+ 1100
 10011
```

Hierbei tritt gleichzeitig ein Übertrag auf (Carry-Ausgang): die ganz links in der Summe stehende 1. Wenn bei diesem Beispiel der Carry-Eingang "1" gewesen wäre, dann sähe das Ergebnis so aus:

```
  0111
  1100
+    1
 10100
```

Die vollständige Liste der möglichen Funktionen wurde bereits in Teil 3 angegeben, ihnen sind die Adressen 7700 . . . 7757 zugeordnet. Die Signale der Steuereingänge sind in diesen Adressen enthalten. Binär lauten die Adressen:
111 111 000 000 . . . 111 111 101 111, wobei die letzten 6 Bit den Steuereingängen M, $\overline{C}$, $\overline{S3}$, $\overline{S2}$, $\overline{S1}$ und $\overline{S0}$ entsprechen. Eine Adresse, die den ALU betrifft, lautet also:
111 111 M $\overline{C}$ $\overline{S3}$ $\overline{S2}$ $\overline{S1}$ $\overline{S0}$.
Einige Bemerkungen:
1. Carry-Ein- und Ausgang sind beim 74181 invertierend. In der Schaltung müssen deshalb $\overline{CI}$ (Carry In) und $\overline{CO}$ (Carry Out), in der Rechnung dagegen CI und CO berücksichtigt werden. In dem erwähnten zweiten Beispiel ist C1 = 1 und CO = 1, an den Anschlüssen des 74181 liegen jedoch in diesem Fall Null Volt.
2. Nicht alle Variationen von M und $\overline{CI}$ gehören zu ALU-Adressen: Adressen mit 111 111 11X XXX sind für andere Aufgaben reserviert (Register, Multiplikation, Division u.a.). Es sind dies die Adressen 7760 . . . 7777.

Rechenregister

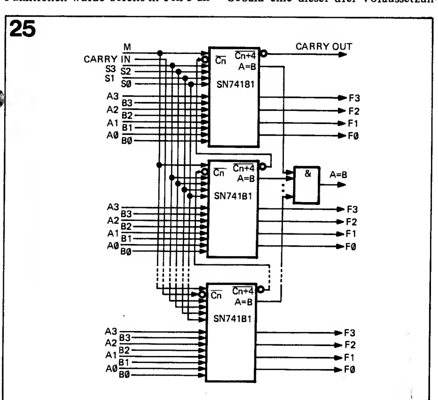
Der ALU SN 74181 enthält keine Speicherelemente. Die gewählte Funktion erscheint an den F-Ausgängen, solange
- die Signale an den Steuereingängen der gewählten Funktion entsprechen und
- an den A-Eingängen die Zahl A und
- an den B-Eingängen die Zahl B liegt.

Sobald eine dieser drei Voraussetzungen wegfällt, ändert sich auch die Information an den F-Ausgängen.
Dies führt beim "Computer 74" zu Schwierigkeiten. Wenn die Funktion anhand der Adresse (SRC oder DST) bestimmt wurde, bleiben nur die Data-Leitungen übrig, um den ALU mit Informationen (z.B. Zahl A) zu versorgen. Für die zweite Zahl (B) ist keine Möglichkeit mehr vorhanden. Selbst wenn diese Möglichkeit gegeben wäre und die gewünschte Funktion gewählt werden könnte, dann würde diese Information doch nicht lange genug vorhanden sein. Daher müßte sie, am besten über die DATA-Leitungen, einer anderen Einheit (z.B. Speicher) zugeführt werden.
Zur Lösung dieses Problems wird ein "Rechenregister" benutzt, das vorläufig mit RR bezeichnet werden soll. Das Zusammenwirken von RR und ALU muß dann folgende Bedingungen erfüllen:
- die Zahl A für den ALU wird stets dem RR entnommen;
- die Zahl B für den ALU läuft über die DATA-Leitungen;
- die Funktion F, ausgeführt mit A und B, wird in das RR geschrieben. Zahl A wird also gelöscht und geht verloren!
- der ALU wird ausschließlich als DST addressiert;
- das RR muß unabhängig vom ALU als SRC oder DST erreichbar sein.

Als Beispiel ein Programm für die Addition der beiden Zahlen P und Q:

(SRC)	START	Speicher M
(DST)		RR
(SRC)		Speicher N
(DST)		OP (nach Ausführung steht im RR: P + Q)
(SRC)		RR
(DST)		Speicher K
(SRC)		HALT (das Ergebnis der Addition (P + Q) befindet sich in Speicher K).
	K,	0
	M,	Zahl P
	N,	Zahl Q

In der Liste der Funktionen aus Teil 3 kommt dieses Zusammenwirken von RR und ALU bereits zum Ausdruck: Zahl A ist darin durch RR und Zahl B durch DATA ersetzt: die Information auf den DATA-Leitungen.
Der gewählte Aufbau bringt einen kleinen Nachteil mit sich. Es gibt nämlich sowohl Funktionen, die sich ausschließlich auf Zahl A beziehen, als auch solche, die den Inhalt des RR betreffen, zum Beispiel: Dekrement



(7700), Linksschieben (7703), Linksschieben und Inkrement (7723) u.a. Es wäre vorteilhaft – im Hinblick auf die Programmierung –, wenn diese Funktionen die Zahl A direkt von den DATA-Leitungen holen würden. Nun muß vor der Ausführung z.B. eines Schiebebefehls, zuerst Zahl A in das RR gebracht (SRC = Zahl A, DST = RR) und danach der Schiebebefehl gegeben werden (SRC = XXX – spielt keine Rolle, muß aber eine bestehende Einheit sein, z.B. Zahl A, ein Speicherplatz also – DST = 7703). Das kostet zwei Befehlsschritte anstelle eines einzelnen.
Das Vertauschen von A und B würde diesen Nachteil beseitigen. Dies ist auch in der Tat möglich! Der Multiplizierer und der Dividierer, die auch vom ALU Gebrauch machen, können jedoch bei der gewählten Lösung ein gutes Stück einfacher ausfallen. Wie noch zu sehen sein wird, benutzen beide Schaltungen intensiv das Linksschieben des RR-Inhaltes. Wenn Multiplizierer und Dividierer weggelassen werden (Multiplikation und Division lassen sich leicht programmieren!), kann die genannte Vertauschung ohne weiteres vorgenommen werden. Die Subtraktion erfolgt dann so: RR: DATA minus RR.
Unter den möglichen Funktionen befinden sich einige, die keinen Sinn haben, z.B.: 7717, 7720, 7740: RR:=RR, auch sind Funktionen doppelt vorhanden wie 7716 und 7741: OR, 7724 und 7744 : AND und andere. Dies ist in der Programmierung des SN 74181 eingebaut, und da jeder Funktionscode eine Adresse ist, kommt dies auch in der Adressenliste zum Ausdruck. Der Gewinn, der zu erreichen wäre, wenn diese Adressen für andere Einheiten benutzt würden, wiegt jedoch die dann erforderliche komplizierte Adressierung des ALU nicht auf.

Carry Out-Register und Gleich-Register

Die Information am Carry-Ausgang des letzten 74181 (Bild 25) wird im Carry Out-Register (CO), bestehend aus einem Flipflop, gespeichert. Auch die "Gleich"-Ausgänge (A=B) enden in einem Flipflop: dem Gleich-Register (GL). Das Carry-Register wird jedesmal abgefragt, wenn ALU oder GL als DST dienen (also wenn DST zwischen 7700 und 7757 liegt bzw. 7766 ist). Das GL-Register wird abgefragt, wenn GL als DST angewiesen ist.
CO wird im Programm ausschließlich als SRC benutzt; als DST hat es keinen Sinn, denn die Information, die im Flipflop eingespeichert wurde, wird überhaupt nicht benötigt. GL kann dagegen DST werden (und natürlich auch SRC).
Als DST arbeitet GL als Vergleicher: Die Information der SRC, die auf den DATA-Leitungen steht, wird mit dem Inhalt des Rechenregisters verglichen. Sind beide gleich, dann wird das GL-Flipflop "1", sonst "0". Ist A (Inhalt RR) > B, dann wird CO = "1", ist A ⩽ B, dann wird CO = 0. Wenn CO oder GL als SRC angewiesen werden, wird der Inhalt des betreffenden Flipflops auf die DATA-Leitung DATA 01 gegeben, das heißt also: 0 oder 2.
Warum DATA 01 und nicht DATA 00? CO und GL werden im Programm zur Entscheidung benötigt, ob eine Zahl A größer, kleiner oder gleich einer Zahl B ist. Hierzu wird A mit B (GL) verglichen, das Ergebnis ist in CO und GL zu finden. Mit GL kann bestimmt werden, ob A = B, mit CO, ob A > B und damit auch, ob A ⩽ B ist. Das Ergebnis des Vergleichs kann nun direkt dazu benutzt werden, einen Sprung auszulösen. Beispiel:

(SRC)	Zahl A
(DST)	RR
(SRC)	Zahl B
(DST)	GL (CO ist jetzt 0, wenn A ⩽ B, 1 wenn A > B)
(SRC)	CO
(DST)	RR
(SRC)	RR
(DST)	OP (Der Inhalt der "Adresse" (=Fortsetzung) wird zu CO addiert: wenn A ⩽ B steht "Fortsetzung" (+0) im RR, wenn A > B steht dort "Fortsetzung" +2.)
(SRC)	Adresse
(DST)	PC (Diese Adresse, also "Fortsetzung" oder "Fortsetzung" +2 wird in den Program Counter gesetzt.)
(SRC) Fortsetzung,	Sprung 1
(DST)	PC
	. . . (hier folgt das Programm für A > B.)
	.
	.
	.
(SRC) Sprung	und so weiter (hier folgt das Programm für A ⩽ B.)
	.
	.
	.
Adresse,	Fortsetzung
Zahl A,	. . .
Zahl B,	. . .
Sprung 1,	Sprung

Wie dieses Beispiel zeigt, kann bei A > B der Inhalt von CO (=2) verwendet werden, um sowohl die SRC- als auch die DST-Adresse des Sprungbefehls ("Fortsetzung, Sprung 1") zu überschlagen.

Negative Zahlen

Die Subtraktion ist definitionsgemäß identisch mit der Addition von negativen Zahlen. Nach genau diesem Prinzip verfährt auch der ALU bei der Subtraktion.
Wie aber sehen negative Zahlen aus? Das kann unterschiedlich sein, es richtet sich nach der vereinbarten Definition. Für binäre Zahlen ist jedoch das sogenannte Zweierkomplement (two's complement) allgemein gebräuchlich.
Zahlen von 12 Bit können dezimal zwischen 0 und 4095 liegen. Größere Zahlen sind nicht darstellbar, kleinere aber auch nicht! Die Vereinbarung für das 2er-Komplement besagt nun, daß die 4096 Zahlen in positive und negative Zahlen aufgeteilt werden, und zwar so, daß hiermit bequem gerechnet werden kann. Das Subtrahieren muß zum Beispiel durch das Addieren einer negativen Zahl möglich sein. Die Unterteilung ist in Tabelle I angegeben.

Tabelle I. Negative Zahlen

dezimal	oktal *)	binär	
+2047	3777	011 111 111 111	
+2046	3776	011 111 111 110	
+2045	3775	011 111 111 101	
.	.	.	po-
.	.	.	si-
.	.	.	tiv
+ 3	0003	000 000 000 011	
+ 2	0002	000 000 000 010	
+ 1	0001	000 000 000 001	
0	0000	000 000 000 000	
– 1	7777	111 111 111 111	
– 2	7776	111 111 111 110	
– 3	7775	111 111 111 101	
.	.	.	ne-
.	.	.	ga-
.	.	.	tiv
–2046	4002	100 000 000 010	
–2047	4001	100 000 000 001	
–2048	4000	100 000 000 000	

*) Siehe unter "Multiplikation"

Um zu einer Zahl (binär) das 2er-Komplement zu finden, wird zuerst das 1er-Komplement (one's complement) gebildet. Das ist einfach: jede 1 wird 0 und jede 0 wird 1. Das 2er-Komplement entsteht daraus durch die Addition vom 1. Beispiel:

	dezimal	oktal	binär
	726	1326	001 011 010 110
1er-Komplement	–	6451	110 100 101 001
		1 +	1 +
2er-Komplement	(–726)	6452	110 100 101 010

Die Subtraktion bereitet nun keine Schwierigkeiten mehr.
Beispiel: 1250 - 726 = 524

```
dezimal    oktal     binär
 1250       2342     010 011 100 010
 -726 +     6452 +   110 100 101 010 +
  524     1 1014   1 001 000 001 100
            carry!
```

Der Carry wird nicht benötigt, er zeigt jedoch an, ob das Ergebnis positiv oder negativ zu werten ist. Wenn zum Beispiel 726 - 1250 = –524 berechnet wird, ist der Carry 0:

```
dezimal    oktal     binär
   726      1326     001 011 010 110
 –1250 +    5436 +.  101 100 011 110 +
 –  524   0 6764   0 110 111 110 100
            carry=0
```

Das Ergebnis ist wieder das 2er-Komplement von 524:

```
                        dezimal  oktal   binär
                          524    1014    001 000 001 100
1er-Komplement             –     6763    110 111 110 011
                                    1 +                1 +
2er-Komplement (–524)            6764    110 111 110 100
```

Bei der Subtraktion zeigt also Carry = 0 an, daß das Ergebnis negativ ist, und Carry = 1, daß das Ergebnis positiv oder 0 ist.
Nachbemerkung: Beim Vergleich von A und B mit GL (siehe oben) arbeitet der ALU in einem Subtraktionsmodus, der im 1er-Komplement subtrahiert. Daher wird in diesem Fall CO = 1, wenn A $\leqslant$ B und CO = 0, wenn A > B.
Es sei noch darauf hingewiesen, daß weder bei oktalen noch bei binären negativen Zahlen ein besonderes Zeichen benötigt wird. Der Unterschied ist unmittelbar an dem "meist signifikanten Bit" (MSB) zu erkennen: dieses ist 0 für die nicht negativen Zahlen (positiv und 0) und 1 für die negativen Zahlen (siehe Tabelle I).

Multiplikation

Vor der Schaltungsbetrachtung für die ALU's soll zunächst beschrieben werden, nach welchem Prinzip die Multiplikation vor sich geht. Die Schaltungen sind in ihrem Aufbau nämlich untereinander sehr ähnlich.
Multiplikation und Division erfolgen, wie alle arithmetischen Operationen mit dem ALU, im binären Zahlensystem, – mit der Basis 2. Eine vereinfachte Schreibweise hierfür ist die bereits benutzte Darstellung im Oktalsystem mit der Basis 8; das Rechenprinzip ändert sich dadurch nicht. Der vorangegangene Text enthält genügend Beispiele.
Die Durchführung der Multiplikation läßt sich in mehrere einfache Schritte unterteilen:

– multiplizieren mit einer Ziffer (0 . . . 9)
– multiplizieren mit 10; dies entspricht der Verschiebung um eine Stelle nach links (eventuell muß eine 0 angehängt werden)
– Addition der Teilergebnisse.

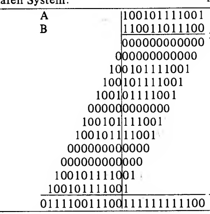
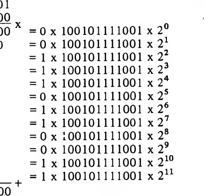
Zur Verdeutlichung folgendes Beispiel:

```
A          3292
B          2425 x
          16460    = 5 x 3292
          6584     = 2 x 3292 x 10
        13168      = 4 x 3292 x 10 x 10
        6584    +  = 2 x 3292 x 10 x 10 x 10
A x B   7983100
```

Dies stellt überhaupt nicht neues dar! Neu ist jedoch, daß die Multiplikation von binären Zahlen auf genau die gleiche Weise durchgeführt werden kann. Die einzelnen Schritte werden dabei noch einfacher:

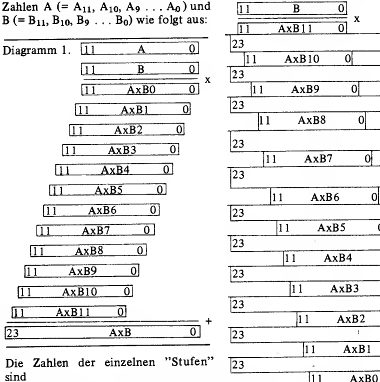
– multiplizieren mit den Ziffern 0 oder 1, das bedeutet nur so viel wie ja oder nein!
– multiplizieren mit 2, dies entspricht auch hier der Verschiebung um eine Stelle nach links.
– Addition der Teilergebnisse.

Das obenstehende Beispiel wird im binären System:

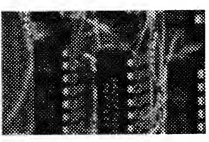
```
A                       100101111001
B                       110011011100   x
                        000000000000     = 0 x 100101111001 x 2^0
                       0000000000000     = 0 x 100101111001 x 2^1
                      100101111001       = 1 x 100101111001 x 2^2
                     100101111001        = 1 x 100101111001 x 2^3
                    100101111001         = 1 x 100101111001 x 2^4
                   000000000000          = 0 x 100101111001 x 2^5
                  100101111001           = 1 x 100101111001 x 2^6
                 100101111001            = 1 x 100101111001 x 2^7
                000000000000             = 0 x 100101111001 x 2^8
               000000000000              = 0 x 100101111001 x 2^9
              100101111001               = 1 x 100101111001 x 2^10
             100101111001             +  = 1 x 100101111001 x 2^11
             011110011100111111111100
```

In der allgemeinen Form sieht die binäre Multiplikation von zwei Zahlen A (= A_{11}, A_{10}, A_9 . . . A_0) und B (= B_{11}, B_{10}, B_9 . . . B_0) wie folgt aus:



Die Zahlen der einzelnen "Stufen" sind

– gleich A (100 101 111 001), wenn $B_i = 1$,

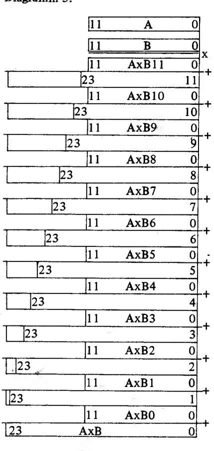


– gleich 0 (000 000 000 000), wenn $B_i = 0$.

Für das Ergebnis ist bedeutungslos, ob alle zwölf Zahlen A · B_i auf einmal addiert werden, oder ob jedesmal ein Zwischenergebnis gebildet wird. Auch die Reihenfolge ist ohne Bedeutung, dagegen muß der Stellenwert natürlich beachtet werden. So stellt auch das nachfolgende Schema die allgemeine Multiplikation von A und B dar:

Das Ergebnis ändert sich auch dann nicht, wenn A • B_i nicht nach rechts, sondern das Zwischenergebnis jedesmal nach links verschoben wird, also so:

Diagramm 3.



Die beschriebene Methode zur binären Multiplikation von zwei Zahlen bietet sich für die Realisation mit dem ALU geradezu an. Sowohl die Addition als auch das Linksschieben lassen sich damit durchführen. Es müssen allerdings Zahlen verarbeitet werden können, die zweimal so lang sind wie für A und B zugelassen; wenn A und B je 12 Bit haben, sind dies 24 Bit. Die "arithmetic and logic unit" muß für 24 Bit ausgelegt werden. Das bedeutet zugleich, daß RR aus zwei (zusammengeschalteten) Registern besteht: RØ für die niedrigeren Stellen, R1 für die höheren, jedes 12 Bit lang.

Die Einheit arbeitet dann wie folgt:

– Zahl B wird in ein Hilfsregister R2 gesetzt. Dies ist ein Schieberegister mit 12 Bit, das nach links schieben kann (also zuerst das signifikanteste Bit B11),
– das endgültige (und zwischenzeitliche) Ergebnis wird in R1 und RØ (RR) gespeichert,
– die Multiplikation startet, wenn Zahl A als SRC mit VM (Multiplizierer) als DST verbunden wird. Zahl A liegt dann an den DATA-Leitungen, sie ist die erste Zahl für den ALU (siehe vorangegangene Beschreibung: b-Eingänge). Die andere Zahl kommt vom RR (a-Eingang, wie schon besprochen).

Die Multiplikation besteht aus 12 Schritten, wobei pro Schritt:

1. der Inhalt des RR eine Stelle nach links schiebt; vor dem ersten Schritt wird RR 0 gesetzt (Reset);
2. abhängig von der Information am Ausgang von R2 (das ist B_i!) das

Diagramm 4.

11 DATA 00

Zahl A → 100101111001

Zahl B → R2

11 ... R2 ... 0		11 ... R1 ... 0	11 ... RØ ... 0	
1¦10011011100	1xA	000000000000	000000000000	*) Reset
			100101111001	+ Schritt 1
1¦00110111001	1xA	000000000000	100101111001	
		000000000001	001011110010	
			100101111001	+ Schritt 2
0¦01101110011	0xA	000000000001	110001101011	
		000000000011	100011010110	
			000000000000	+ Schritt 3
0¦11011100110	0xA	000000000011	100011010110	
		000000000111	000110101100	
			000000000000	+ Schritt 4
1¦10111001100	1xA	000000000111	000110101100	
		000000001110	001101011000	
			100101111001	+ Schritt 5
1¦01110011001	1xA	000000001110	110011010001	
		000000011101	100110100010	
			100101111001	+ Schritt 6
0¦11100110011	0xA	000000011110	001100011011	
		000000111100	011000110110	
			000000000000	+ Schritt 7
1¦11001100110	1xA	000000111100	011000110110	
		000001111000	110001101100	
			100101111001	+ Schritt 8
1¦10011001101	1xA	000001111001	010111100101	
		000011110010	101111001010	
			100101111001	+ Schritt 9
1¦00110011011	1xA	000011110011	010101000011	
		000111100110	101010000110	
			100101111001	+ Schritt 10
0¦01100110111	0xA	000111100111	001111111111	
		001111001110	011111111110	
			000000000000	+ Schritt 11
0¦11001101110	0xA	001111001110	011111111110	
		011110011100	111111111100	
			000000000000	+ Schritt 12
1¦10011011100		011110011100	111111111100	

*) ◄—: links schieben

DATA-Signal (Zahl A) entweder zu RR addiert wird (wenn $B_i = 1$), oder auch nicht (wenn $B_i = 0$);

3. der Inhalt von R2 eine Stelle nach links schiebt, dadurch gelangt das nächste Bit von Zahl B zum Ausgang.

Nach 12 solchen Schritten steht das Ergebnis in RR.
Der Ausgang von R2 kann zum Eingang rückgekoppelt werden, so daß der Inhalt nicht verlorengeht. Damit ist dann allerdings die frühere Information von RR gelöscht!

Als Beispiel wurde in Diagramm 4 die obenstehende Multiplikation 3292 • 2425 = 7 983 100 vollständig durchgeführt.

Programmierung: (SRC) Zahl B, (DST) R2; (SRC) Zahl A, (DST) VM. Das Ergebnis steht in R1 & RØ.

Diese Art der Multiplikation eignet sich nicht für negative Zahlen. Die Zahlen A und B werden als "unsigned integer" aufgefaßt: ganze Zahlen ohne Vorzeichen (also positiv), von 0 bis 4095 laufend. Wenn trotzdem "signed integers" (positiv und negativ, von −2048 bis +2047) verarbeitet werden sollen, müssen die negativen Zahlen in positive umgewandelt werden (von 0 abziehen!), das Vorzeichen des Produktes muß dann getrennt bestimmt werden,

Tabelle I gibt an, daß dem Pluszeichen eine 0 an der signifikantesten Stelle (ganz links) der binären Zahl und dem Minuszeichen eine 1 an dieser Stelle entspricht. Bekanntlich ist das Produkt von zwei Zahlen positiv, wenn beide Zahlen gleiches Vorzeichen haben, bei unterschiedlichem Vorzeichen dagegen negativ.
Analog dazu gilt für binäre Zahlen: Ergebnis ist Plus, wenn beide MSB's (Most Significant Bits) 0 oder beide 1 sind; Minus, wenn das eine 0 und das andere 1 ist. Wird das Vorzeichen des Produktes auch durch 0 bzw. 1 ausgedrückt, dann liefert dies die Exclusive-OR-Funktion aus den Vorzeichen beider Zahlen. $MSB_a = 0$ und $MSB_b = 0$, oder $MSB_a = 1$ und $MSB_b = 1$, dann $MSB_{Produkt} = 0$; $MSB_a = 0$ und $MSB_b = 1$ oder umgekehrt, dann $MSB_{Produkt} = 1$.

Division

Ebenso wie die Multiplikation läßt sich auch die Division in mehrere einfachere Schritte zerlegen. Die Division ist im Prinzip die Umkehrung der Multiplikation. Schieberichtung der Multiplikation (nach links) und anschließende Addition kehren sich in der Tat um: Rechtsschieben und Subtraktion sind die Schritte bei der Division.
Bei der Multiplikation von A und B wurde die Zahl A, abhängig von $B_i = 0$ oder 1, addiert bzw. nicht addiert; bei der Division hängt dagegen der Wert B_i (0 oder 1) davon ab, ob A subtrahiert werden kann oder nicht. Auch für die Division läßt sich ein allgemeines Schema aufstellen:

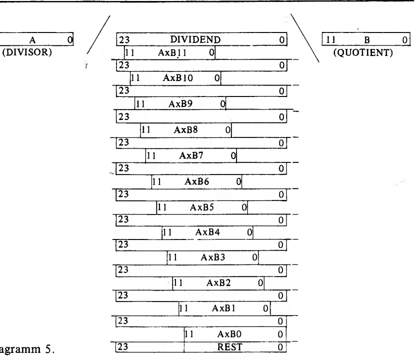


Diagramm 5.

Bei der Betrachtung des Schema's (Diagramm 5) fällt folgendes auf:
– die Zahl A • B_i schiebt "von Natur aus" nach rechts, was bei der Multiplikation erst nach Änderung der Ziffernfolge der Fall war,
– ein Zwischenergebnis benutzt man bei der Division auch normalerweise, denn die gleichzeitige Subtraktion von mehreren Zahlen ist recht mühsam.

Für die Durchführung der Division mit ALU's wird das Rechtsschieben von A • B_i wieder durch das Linksschieben des Zwischenergebnisses ersetzt:

11 A 0 | 23 DIVIDEND 0 | 11 AxB11 0 | 22 0 | 11 AxB10 0 | 21 0 | 11 AxB9 0 | 20 0 | 11 AxB8 0 | 19 0 | 11 AxB7 0 | 18 0 | 11 AxB6 0 | 17 0 | 11 AxB5 0 | 16 0 | 11 AxB4 0 | 15 0 | 11 AxB3 0 | 14 0 | 11 AxB2 0 | 13 0 | 11 AxB1 0 | 12 0 | 11 AxB0 0 | 11 REST 0 | 11 B 0

Diagramm 6.

Die Vorgänge bei der Division mit dem ALU sind zu denen der Multiplikation komplementär: Der Dividend steht im Rechenregister (R1 & RØ), der Divisor (Zahl A) auf den DATA-Leitungen, das Ergebnis, also der Quotient (Zahl B), kommt in R2. Der eventuell entstehende Rest steht in R1, dem höherwertigen Teil von RR (siehe Diagramm). Auch hier sind wieder 12 Schritte zu unterscheiden, wobei pro Schritt:

1. der Inhalt von RR (Dividend) eine Stelle nach links schiebt;
2. B_i 0 oder 1 wird und demzufolge A subtrahiert wird oder nicht subtrahiert wird, dies hängt davon ab, ob A subtrahiert werden kann ($B_i = 0$) oder nicht ($B_i = 1$);
3. B_i in R2 geschoben wird (nach links).

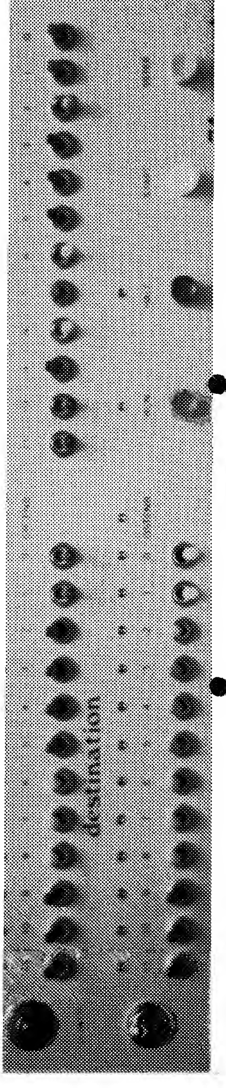
Auch hier ist der Vorgang nach 12 Schritten beendet.

Es folgt nun in Diagramm 7 ein ausführliches Beispiel, die Division 8 885 774 : 2418 = 3674, Rest 2042:

Diagramm 7.

11 ... R2 ... 0	Zahl B		11 .. DATA .. 00 / 11 ... R1 ... 0	11 ... RØ ... 0	
			100101110010		← Zahl A (Divisor)
xxxxxxxxxxxx					
			100001111001	011000001110	← Dividend
		1	000011110010	110000011100	
xxxxxxxxxxx1	← 1xA		100101110010		− Schritt 1
			011110000000	110000011100	←
		0	111100000001	100000111000	
xxxxxxxxxx11	← 1xA		100101110010		− Schritt 2
			010110001111	100000111000	←
		0	101100011111	000001110000	
xxxxxxxxx111	← 1xA		100101110010		− Schritt 3
			000110101101	000001110000	←
		0	001101011010	000011100000	
xxxxxxxx1110	← 0xA		000000000000		− Schritt 4
			001101011010	000011100000	←
		0	011010110100	000111000000	
xxxxxxx11100	← 0xA		000000000000		− Schritt 5
			011010110100	000111000000	←
		0	110101101000	001110000000	
xxxxxx111001	← 1xA		100101110010		− Schritt 6
			001111110110	001110000000	←
		0	011111101100	011100000000	
xxxxx1110010	← 0xA		000000000000		− Schritt 7
			011111101100	011100000000	←
		0	111111011000	111000000000	
xxxx11100101	← 1xA		100101110010		− Schritt 8
			011001100110	111000000000	←
		0	110011001101	110000000000	
xxx111001011	← 1xA		100101110010		− Schritt 9
			001101011011	110000000000	←
		0	011010110111	100000000000	
xx1110010110	← 0xA		000000000000		− Schritt 10
			011010110111	100000000000	←
		0	110101101111	000000000000	
x11100101101	← 1xA		100101110010		− Schritt 11
			001111111101	000000000000	←
		0	011111111010	000000000000	
111001011010 (Quotient)	← 0xA		000000000000		− Schritt 12
			011111111010 (Rest)	000000000000	

Programmierung: (SRC) Dividend (höherwertige 12 Bit), (DST) R1; (SRC) Dividend (niederwertige 12 Bit), (DST)



RØ; (SRC) Zahl A, (DST) DL. Das Ergebnis steht in R2, der Rest steht in R1.
Im Gegensatz zur Multiplikation taucht bei der Division eine Schwierigkeit auf: Das Produkt von zwei Zahlen von je maximal 12 Bit kann niemals länger als 24 Bit werden. Dagegen ist es durchaus möglich, daß der Quotient aus einer Zahl von 24 Bit und einer Zahl von maximal 12 Bit länger als 12 Bit wird. Die Länge des Ergebnisses B könnte also unter Umständen die Kapazität von R2 übersteigen!
Schon ein ganz einfaches Beispiel macht dies deutlich: Wenn A = 2 und der Dividend > 8192 ist, so wird B > 4096 und damit länger als 12 Bit. Bei der Subtraktion und Verschiebung tritt dann ein "Overflow" auf, so daß das Ergebnis nicht mehr exakt sein kann. Dies muß zumindest angezeigt angezeigt werden.
Es besteht jedoch eine Möglichkeit, mit nur wenig zusätzlichem Aufwand volle Genauigkeit zu erreichen: Division einer Zahl von maximal 24 Bit durch eine Zahl von maximal 12 Bit, mit einem Ergebnis von maximal 24 Bit und einem Rest von maximal 12 Bit (die größte mit 24 Bit darstellbare Zahl ist (dezimal) 16 777 215!).
Hierzu soll Diagramm 5, das allgemeine Schema für die Division, noch einmal kurz betrachtet werden. In dem Fall, daß das Ergebnis B ⩽ 4095 ist, besteht der Rest aus weniger als 12 Bit. In obengenanntem Fall ist jedoch nicht jede Subtraktion imstande, den höherwertigen Teil des Restes zu "beseitigen", es werden Bits übrig bleiben. Beispiel:

Der Rest, der auf diese Weise übrig bleibt, ist noch durch A teilbar. Das bedeutet natürlich, daß der höherwertige Teil des Restes verfügbar sein muß. Bei der Ausführung mit RR und ALU ist dies nicht ohne weiteres der Fall. Wie aus Diagramm 7 hervorgeht, kann RØ hierzu herangezogen werden. Die "aufzubewahrenden" Bits könnten dann mit dem schrittweisen Schieben in RØ eingespeichert werden.
Leider stehen die Overflow-Bits erst zur Verfügung, wenn der Schiebevorgang beendet ist. Der ganze Zyklus von 12 Schritten müßte also anders aufgebaut werden (ähnlich wie bei der Multiplikation). Aus diesem Grund läßt sich hier besser ein viertes Register R3 einsetzen. R3 ist ebenso wie R2 ein Schieberegister, in das bei jedem Schritt ein Overflow-Bit eingespeichert wird.
Die Programmierung der Division sieht nun wie folgt aus: Der Dividend wird in RR (R1 & RØ) eingespeichert, der Divisor steht auf den DATA-Leitungen; Ausführung der Division (siehe oben). Dann wird kontrolliert, ob R3 0 ist. Wenn nicht, wird zuerst der Inhalt von R2 abgespeichert, dann der REST (R1) in RØ gesetzt, R3 in R1, und die Division durch A wiederholt. Das Ergebnis der neuen Division (in R2) wird zu dem ersten Ergebnis addiert. Dies kann, falls erforderlich, so oft wiederholt werden, bis R3 0 bleibt. Die Summe der "Teil"-Quotienten bildet das endgültige Ergebnis, der REST ist der zuletzt gefundene Rest.
Im folgenden Beispiel ist eine Division mit "Overflow" vollständig dargestellt:
10 078 983 : 2418 =
4059 Rest 262 144 =
4059 + (262 144 : 2418) =
4059 + 109 Rest 759 =
4168 Rest 759

```
A | DIVIDEND | 1111 ... B
  AxB11
  AxB10
  AxB9
  AxB8
  1
  AxB7
  1
  AxB6
  1
  AxB5
  1
  AxB4
  1 1
  AxB3
  1 1
  AxB2
  1 1
  AxB1
  1 1
  AxB0
  1 1 REST
```

Diagramm 8.

```
11 ... R3 ... 0    11 ... R2 ... 0       11 .. DATA ..00
xxxxxxxxxxxx       xxxxxxxxxxxx           100101110010   (Divisor)

                                          11 ... R1 ... 0 | 11 ... RØ ... 0
                                            100110011100  101100000111  (Dividend)
                                          1 001100111001  011000001110
                   xxxxxxxxxxx1 <- 1xA      100101110010                 -
xxxxxxxxxxx0 <- 0                           100111000111  011000001110
                                          1 001110001110  110000011100
                   xxxxxxxxxx11 <- 1xA      100101110010                 -
xxxxxxxxxx00 <- 0                           101000011100  110000011100
                                          1 010000111001  100000111000
                   xxxxxxxxx111 <- 1xA      100101110010                 -
xxxxxxxxx000 <- 0                           101011000111  100000111000
                                          1 010110001111  000001110000
                   xxxxxxxx1111 <- 1xA      100101110010                 -
xxxxxxxx0000 <- 0                           110000011101  000001110000
                                          1 100000111010  000011100000
                   xxxxxxx11111 <- 1xA      100101110010                 -
xxxxxxx00000 <- 0                           111011001000  000011100000
                                          1 110110010000  000111000000
                   xxxxxx111111 <- 1xA      100101110010                 -
xxxxxx000001 <- 1                     <-  1 010000011110  000111000000
                                          0 100000111100  001110000000
                   xxxxx1111110 <- 0xA      000000000000                 -
xxxxx0000010 <- 0                           100000111100  001110000000
                                          1 000001111000  011100000000
                   xxxx11111101 <- 1xA      100101110010                 -
xxxx00000100 <- 0                           011100000110  011100000000
                                          0 111000001100  111000000000
                   xxx111111011 <- 1xA      100101110010                 -
xxx000001000 <- 0                           010010011010  111000000000
                                          0 100100110101  110000000000
                   xx1111110110 <- 0xA      000000000000                 -
xx0000010000 <- 0                           100100110101  110000000000
                                          1 001001101011  100000000000
                   x11111101101 <- 1xA      100101110010                 -
x00000100000 <- 0                           100011111001  100000000000
                                          1 000111110011  000000000000
                   111111011011 <- 1xA      100101110010                 -
000001000000 <- 0                           100010000001  000000000000
(höherer Rest)     (erster Quotient)        (niedriger Rest)
```

Diagramm 9.

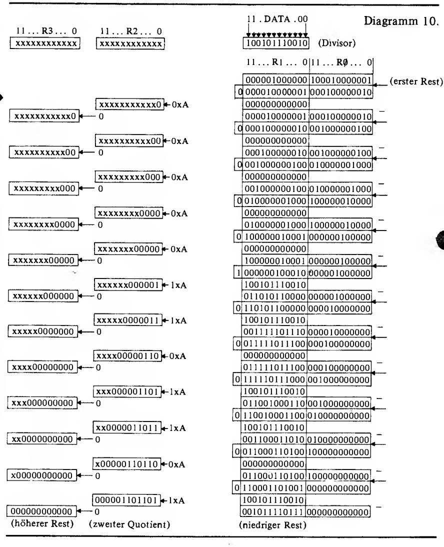
Tebelle II. Programm für die Division in doppelter Zehlenlänge.

```
0000 7762  START,   RØ       (setze 0:
0001 7754             ZRO    (RR,
0002 7762           RØ
0003 0055             REST   (Rest,
0004 7762           RØ
0005 0056             QUOTL  (Quotient niedrig
0006 7762           RØ
0007 0057             QUOTH  (Quotient hoch;
0010 0060  TEILE,   DEELL    (setze
0011 7762             RØ     (Dividend
0012 0061           DEELH    (in
0013 7763             R1     (RR (R1 und RØ);
0014 0062  TEILE2,  DIVISOR  (teile
0015 7761             DL     (durch Divisor
0016 7763           R1       (hebe R1 euf
0017 0055             REST   (im Rest
0020 0057           QUOTH    (zähle elten Quotient
0021 7763             R1     ((Quoth & Quotl)
0022 0056           QUOTL    (zusemmen
0023 7762             RØ     (mit
0024 7764           R2       (R2 (= neuer Quotient);
0025 7706             OP
0026 7763           R1       (hebe dieses
0027 0057             QUOTH  (Ergebnis
0030 7762           RØ       (wieder auf
0031 0056             QUOTL  (zu Quoth & Quotl;
0032 7762           RØ       (setze RR 0;
0033 7754             ZRO
0034 7765           R3       (vergleiche R3 hiermit:
0035 7766             GL     (ist Overflow 0 ?
0036 7766           GL
0037 7762             RØ     (setze GL-Ergebnis in RØ;
0040 0063                    ADRES1
0041 7706                      OP     (zähle Inhelt von Adres1 dezu
0042 7762                    RØ
0043 7776                      PC     (setze dieses Ergebnis in den PC
0044 0064  SPRUNG,           ADRES2   (R3=0; Division nicht vollständig:
0045 7776                      PC     (springe nach Continu
0046 7777                      HL     (R3 = 0: fertig und stop,
                                      (Ergebnis in Quoth & Quotl.
0047 0055  CONTINU           REST     (bilde neuen Dividend
0050 7762                      RØ     (Rest in RØ
0051 7765                    R3       (R3 nach R1;
0052 7763                    R1
0053 0065                    ADRES3   (und dividiere erneut
0054 7776                      PC     (springe nach Teile2)
0055 0000  REST              0
0056 0000  QUOTL             0
0057 0000  QUOTH             0
0060 5407  TEILL             5407     (niedriger Teil des Dividend
0061 4634  TEILH             4634     (höherer Teil des Dividend
0062 4562  TEILER            4562     (Divisor
0063 0044  ADRES1            SPRUNG   (Hilfsadresse für Sprung
0064 0047  ADRES2            CONTINU  (Hilfsadresse für Continu
0065 0014  ADRES3            TEILE2   (Hilfsadresse für Teile2
```

Nachbemerkung: Durch Hinzufügen von R3 leuten die Adressen der Rechenregister etwas anders els in Teil 3 engegeben:

```
7767 CO   Cerry Out Register (0 oder 2)
7766 GL   Gleich Register (0 oder 2, wenn SRC)
7765 R3   Teil-Overflow Register
7764 R2   Multiplikations/Divisions-Register
7763 R1   RechenRegister 1 (höherer Teil von RR)
7762 RØ   RechenRegister 0 (niedriger Teil von RR)
7761 DL   Division
7760 VM   Multiplikation.
```

Der Inhalt von R3 ist nicht 0, so daß nochmals geteilt werden muß. Erst wird der Inhalt von R2 "sichergestellt", dann der Inhalt von R1 in RØ gesetzt, der von R3 in R1. Nun folgt die zweite Division:



R3 ist jetzt 0: Die Division ist beendet. Das Endergebnis besteht aus dem neuen Inhalt von R2 plus dem vorigen (weggespeicherten) Inhalt. Der Rest ist der letzte Inhalt von R1.

Tabelle II gibt als Beispiel ein vollständiges Programm für die Division an. Die SRC- und DST-Bezeichnungen wurden weggelassen, statt dessen wurden die Zeilen für DST etwas eingerückt. Die Zahlen am Anfang jeder Zeile sind die Speicheradressen, danach folgt der Inhalt (beides oktal).

Der Rest, der nach der Division übrig bleibt, hat eine maximale Länge von 12 Bit. Der Divisor, ebenfalls maximal 12 Bit, kann kleiner sein als der Rest. Es muß dann nochmals geteilt werden. In diesen Fällen reicht die Kontrolle, ob R3 = 0 ist, nicht aus; es muß auch noch untersucht werden, ob der REST kleiner ist als der Divisor. Anschließend wird, falls erforderlich, weiterdividiert. Erst dann kann von einer vollständigen Division die Rede sein.

(wird fortgesetzt)

J.T.W. Damen

Computer 74

Teil 6

Im vorangegangenen Teil 5 wurde die Arbeitsweise der Recheneinheit bei den Operationen Addition, Subtraktion, Multiplikation und Division beschrieben. Diese Folge beschäftigt sich mit der praktischen Schaltungsauslegung.

Die Schaltung

Die wichtigsten Funktionsblöcke der Recheneinheit wurden in einem Blockschaltbild zusammengefaßt (Bild 26). Folgende Blöcke sind zu unterscheiden:

- der ALU, bestehend aus 6 × SN 74181,
- R_ϕ und R_1, bestehend aus je 2 × SN 74174 (6 D-Flipflops),
- R_2 und R_3, bestehend aus je 3 × SN 74179 (4 Bit-Schieberegister),
- die Multiplizier- und Dividierlogik, sie steuert die bereits früher beschriebenen 12 Rechenschritte mit den Operationen Addition, Subtraktion und Schieben,
- der Adressenselektor für alle genannten Funktionsblöcke.

Die Verbindungen der Blöcke stellen Informationsflüsse dar. Der Adressenselektor liefert Steuersignale an:

- VM, nur DST,
- DL, nur DST,
- R_ϕ & R_1 (RR), SRC und DST,
- R_2, SRC und DST,
- R_3, nur SRC,
- CO, nur SRC,
- GL, SRC und DST.

Mit der Multiplizier- und Dividierlogik sind alle vorhandenen Register (außer CO und GL) und der ALU verbunden. Der ALU tauscht seinerseits Informationen mit dem RR aus.

Multiplizier- und Dividierschaltung

Die für die Steuerung von Multiplikation und Division zuständige Schaltung besteht aus einem Clockgenerator (Bild 27), der die einzelnen Schritte auslöst, einem 12-Teiler, der 12 Schritte abzählt, einer Logik, die die Kriterien für die Addition (beim Multiplizieren) bzw. für die Subtraktion (beim Dividieren) prüft, sowie einer Selektionsschaltung für Addieren, Subtrahieren und Schieben. Bild 28 zeigt die vollständige Multiplizier-/Dividierschaltung.

Der aus zwei rückgekoppelten Monoflops KLOKP und KLOKN (SN 74123) bestehende Generator startet, wenn VDDST 1 wird (siehe auch Bild 29). Die Clocksignale werden einerseits in zwei sich abwechselnde Signale SCHUIF und OPAF umgeformt; sie bestimmen die Zeitbereiche, in denen der ALU schiebt oder addiert/subtrahiert. Andererseits werden vom Generatorsignal auch die Clockimpulse für den 12-Teiler VD12DL (SN 7492) abgeleitet. Der Clockgenerator ist so aufgebaut, daß er genau die Anzahl der benötigten Clockimpulse liefert, und zwar ohne störende Resetimpulse im Clocksignal.

Von den Clocksignalen werden zwei weitere Signale abgeleitet: RRSCHF und RROPAF. Diese Impulse liegen zeitlich so, daß in der Mitte des aktiven Bereichs von SCHUIF eine positive Flanke (RRSCHF) und in den aktiven Bereich von OPAF eine negative Flanke (RROPAF) fällt. Dies hat den Zweck,

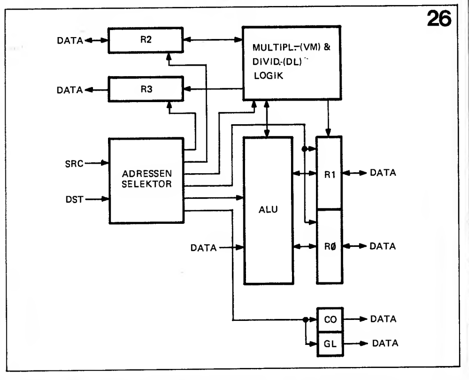


Bild 26. Blockschema der Recheneinheit. Die Multiplizier- und Dividierlogik (VM & DL) ist in Bild 27 als Blockschaltbild und in Bild 28 detailliert angegeben, Bild 29 zeigt die zeitlichen Abläufe. Die Arithmetic and Logic Unit (ALU) und der ALU-Modus-Selektor sind in Bild 30 bzw. Bild 31 dargestellt. Bild 32 zeigt die Rechenregister R_1 & R_ϕ des Multiplikations-/Divisionsregister R_2 ist in Bild 33, das Overflow-Register in Bild 34 angegeben. Schließlich zeigen die Bilder 35 und 36 das Carry-Out- und Gleich-Register bzw. den Adressenselektor.

27

GENERATOR
:12
PRÜF-UND SELEKTIONS LOGIK
R2
R3
SCHIEBEN
ADDIEREN
SUBTRA-HIEREN
ALU
RR

Blockschaltbild Multiplizier/Dividierlogik.

28

KLOKP, KLOKN, SN74123, SCHUIF, SN7474, OPAF, RROPAF, VDDRST, VDDRDY, DSTRDY (BUS), VD12DL, SN7492, PWRCLR (BUS), VDKLRS, KLOKN, KLOKPN, OPAF, SCHUIF, R2KLOK, RR SCHF, VDDST, VDSTRT, (ADRSEL), VDSTOP, VMRRRS, VMRRRS (RR), DLSCKL, VMDST, DLDST, R2OUT (R2), VR2OUT, VDR2IO, VMR2IN, DLR2IN, VDRROA, VDRRKL, VDRRKL (RR), VDR2IN, VDR2IN (R2), DR2IN, SFCRY, VDR2KL, VDR2KL (R2), DLA>B, SN7474, DLR3KL, DLR3KL (R3), VMR2KL, DLR2KL, DLR3IN, DLR3IN (R3), ALUCO (ALU), DLOPAF, VM OP AF, SCHIEBEN, (ALUMOD)

Multiplizier- und Dividierschaltung (VD).

29

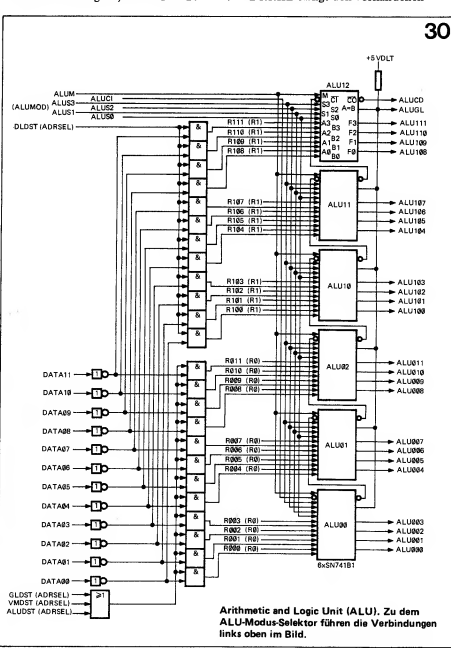
VDDST, VDSTRT, KLOKPN, KLOKN, SCHUIF, OPAF, RRSCHF, RROPAF, VD12DL, VDKLRS, VDDRST, DSTRDY, VMDST, DLDST, "OP", "AF", VMRRRS, DLSCKL, ALUCO, SFCRY, VDR2IN, VDR2KL, R2OUT, DLR3IN, DLR3KL, "R2", "R3", VDRROA, VDRRKL



Zeitdiagramm: Multiplizieren — Dividieren.



daß der ALU jedesmal, wenn SCHUIF aktiv ist ("1"), in den Schiebemodus kommt. Der Inhalt von RR liegt dann um eine Stelle nach links verschoben an den Ausgängen des ALU und daher auch an den Eingängen des gleichen Rechenregisters.
Der Schiebevorgang ist erst vollzogen, wenn die neue Information auch tatsächlich in RR eingelesen wurde, hierzu dient RRSCHF.
RROPAF hat die gleiche Funktion bezüglich der Addition beim Multiplizieren und der Subtraktion beim Dividieren. Bei RROPAF sind die aktiven Flanken negativ. RROPAF muß nämlich noch eine UND-Schaltung passieren, die das "Ja oder Nein" für die Addition bzw. Subtraktion untersucht. Wie aus Bild 29 hervorgeht, wird bei der Multiplikation "OP" und bei der Division "AF" 12mal aktiv. Der ALU arbeitet bei jedem Schritt abwechselnd im Schiebe- und im Zu/Ab-Modus. Die UND-Schaltung sorgt dafür, daß das Clocksignal nur dann zum RR gelangt, wenn tatsächlich addiert oder subtrahiert werden muß. Wie dies vor sich geht, ist aus Bild 28 und 29 zu entnehmen. (Die gewählten Beispiele stimmen mit der Multiplikation aus Diagramm 4 und der Division mit Overflow, 1. Teil, aus Diagramm 9 überein; siehe Teil 5 dieses Artikels).
Zuerst die Multiplikation: R2OUT, das Ausgangssignal von R_2 (B_i!) wird über VR2OUT und VDR2IO im NAND-Gatter VDRROA mit RROPAF verknüpft. Zum Ausgang dieses Gatters gelangen (invertiert!) nur die Impulse von RROPAF, die mit R2OUT zusammenfallen. Die Addition erfolgt daher ausschließlich bei $B_i = 1$.
Wenn $B_i = 0$ ist, ändert sich der Inhalt von RR nicht. Das über VDRROA gewonnene Signal wird mit RRSCHF kombiniert und als Clocksignal den beiden SN 74174 zugeführt, die das RR bilden.
VDR2KL, auch vom Clocksignal des Generators abgeleitet, schiebt R_2 am Ende eines jeden Schrittes um ein Bit weiter, so daß B_{i+1} erscheint.
Zusammenfassung: SCHUIF und OPAF setzen den ALU abwechselnd in den Schiebe- und den Zu/Ab-Modus, VDRRKL bringt den vorhandenen Modus dadurch zur Wirkung, daß die neue Information tatsächlich in das Rechenregister eingelesen wird; VDR2KL gibt das neue B_i frei.
Während der Multiplikation bestimmt B_i, also der Ausgang von R_2, ob addiert werden muß oder nicht. Dagegen muß die Subtraktion stattfinden, wenn subtrahiert werden kann. Das ist der Fall, solange RR ⩾ DATA ist. Am Carry-Ausgangssignal des ALU läßt sich dies leicht erkennen. Dieser befindet sich ja jedes Mal im Subtrahiermodus! Das ist jedoch nicht das einzige Kriterium für die Durchführung der Subtraktion.
Aus den Diagrammen 7, 9 und 10 geht hervor, daß die Subtraktion auch möglich ist, wenn beim vorangegangenen Schieben ein "Overflow" auftrat (das ganz links stehende Bit in den erwähnten Diagrammen). Das Kriterium lautet also: Entweder Subtrahiercarry ist 1 oder Schiebecarry ist 1. Das NAND-Gatter DLA ⩾ B prüft dies. An einem Eingang liegt der abgehende Carry des ALU (Subtrahiercarry), am anderen die Information vom Flipflop SFCRY. Hierin ist der Carry vom vorhergehenden Schieben gespeichert (eingelesen durch DLSCKL, direkt von RRSCHF abgeleitet).
Dadurch, daß der ALU das Carrysignal invertiert liefert (es wird auch invertiert von SFCRY gespeichert), muß für DLA ⩾ B ein ODER-Gatter mit invertierenden Eingängen gewählt werden: $\overline{A \cdot B} = \overline{A} + \overline{B}$ (de Morgan'sches Gesetz). Die Information, die das Gatter liefert, wird zweifach verwendet. Zum einen ist dies B_i, als solches gelangt es zu R_2 (über DLR2IN und VDR2IN, sie unterscheiden Multiplikation und Division) und wird dort gespeichert. Zweitens ist es das Kriterium für Subtrahieren ($B_i = 1$) oder nicht Subtrahieren ($B_i = 0$), der DLA ⩾ B-Ausgang ist daher mit VDRROA verbunden (ebenfalls über Zwischengatter).
Es kommt hier eine mit der Multiplikation übereinstimmende "Impulsausblendung" zustande: Nur die Impulse von RROPAF werden durchgelassen, die mit $B_i = 1$ zusammenfallen, nur unter dieser Voraussetzung wird tatsächlich subtrahiert.
Das Kriterium für Subtrahieren war: entweder der Schiebecarry = 1 oder Subtrahiercarry = 1. Ist sowohl das Subtrahiercarry als auch das Schiebecarry = 1, dann ist das letzte an sich überflüssig. Durch Subtraktion kann dieses "Schiebebit" nicht beseitigt werden. Dies ist vielmehr der Overflow, von dem bei der Division die Rede war, und der im Rest übrig bleibt.
Es ist also das Kriterium dafür, daß eine 1 nach R_3 übertragen werden muß. NOR-Gatter DLR3IN ist dafür verantwortlich. In Übereinstimmung mit DLA ⩾ B arbeitet dieses NOR-Gatter als UND-Gatter: $\overline{A + B} = \overline{A} \cdot \overline{B}$.
R_3 und R_2 erhalten gleichzeitig (während der Division) ein Clocksignal, so daß B_i in R_2 und das Overflow-Bit in R_3 gespeichert werden.
Schließlich spalten zwei UND-Gatter OPAF das Signal in OP (Multiplikation:



Arithmetic and Logic Unit (ALU). Zu dem ALU-Modus-Selektor führen die Verbindungen links oben im Bild.

VMOPAF) und AF (Division: DLOPAF) auf. Diese Signale gelangen zusammen mit SCHUIF zu einer Schaltung, die den Modus des ALU bestimmt.

Arithmetic and Logic Unit

In Bild 25 wurde bereits angedeutet, wie die einzelnen ALU's vom Typ SN 74181 miteinander verbunden werden. Sechs dieser IC's bilden den kompletten ALU für arithmetische und logische Operationen mit einer Kapazität von 24 Bit (Bild 30).
Wie schon früher erwähnt, sind die A-Eingänge mit den Ausgängen des Rechenregisters (R_ϕ & R_1) dauernd verbunden. Die B-Information kommt über die DATA-Leitungen zum ALU. Insgesamt sind 12 DATA-Leitungen vorhanden. Die "untersten" drei ALU's sind mit ihnen in dem Fall verbunden, daß der ALU selbst als Funktionseinheit programmiert wird.
Auch beim Multiplikationszyklus ist dies der Fall (vergleiche Diagramm 4). Außerdem müssen die DATA-Leitungen mit den untersten drei ALU's verbunden werden, wenn GL als Vergleicher (DST) arbeitet. Die drei Signale GLDST, VMDST und ALUDST (vom Adressenselektor) schalten diese Verbindungen durch. Nur bei der Division gelangen die Informationen der DATA-Leitungen zu den "obersten" drei ALU's (siehe Diagramm 7, 9 und 10). In diesem Fall schaltet DLDST durch.

ALU-Modus-Selektor

Die sechs Modus-Eingänge (M, $\overline{C_{in}}$, S_3, S_2, S_1, S_ϕ) des ALU werden von einer aus 6 IC's vom Typ SN 7454 bestehenden Schaltung gesteuert (Bild 31). Im Normalfall, wenn eine der ALU-Funktionen als DST gewählt wird, sind die genannten Eingänge (invertiert oder nicht invertiert) mit den DST-Leitungen DSTϕ5, ϕ4, ϕ3, ϕ2, ϕ1 und ϕϕ verbunden.
Die anderen Modi, in denen der ALU arbeiten kann, sind: Schieben, Addieren und Subtrahieren. Schieben kommt zustande durch SCHUIF, Addieren durch VMOPAF und Subtrahieren durch DLOPAF bei der Division, durch GLDST beim Vergleich. Aus dem 2er-Komplement-Subtraktionsmodus bei der Division wird beim Vergleichen ein 1er-Komplement-Modus.
An den invertierenden Eingängen der SN 7454-IC's kommt dies entsprechend zum Ausdruck: 100 110 (2er-Komplement) und 110 110 (1er-Komplement). GLDST ist daher mit dem ALUCI-Eingang verbunden und legt hieran beim Vergleich eine 1.

Das RechenRegister RR

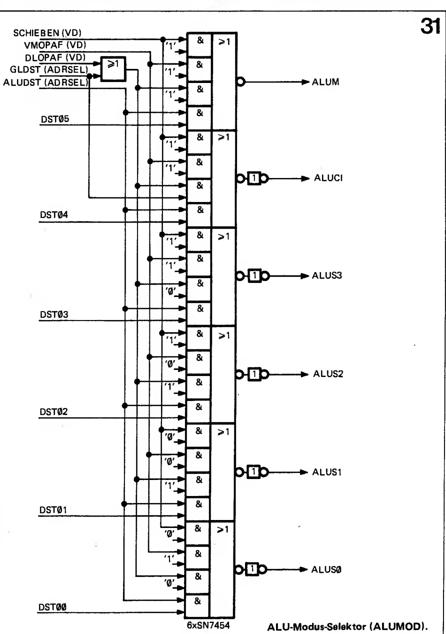
Die Register R_1 und R_ϕ, die zusammen das Rechenregister bilden, bestehen aus IC's vom Typ SN 74174. Diese Register können auf zwei verschiedene Arten mit Informationen versorgt werden (Bild 32). Zum ersten sind R_1 und R_ϕ als DST programmierbar. Die Eingänge sind in diesem Fall mit den DATA-Leitungen verbunden, R1DST bzw. RϕDST bewirken dies. Die Clockeingänge erhalten dabei einen von R1DST und RϕDST abgeleiteten Clockimpuls: R1DSTK und RϕDSTK.

Im zweiten Fall muß die Information des ALU gespeichert werden. Die Ausgänge des ALU müssen dann mit dem Register verbunden werden, außerdem ist ein geeigneter Clockimpuls erforderlich.

Was das erste betrifft: Wenn R_1 und R_ϕ nicht als DST auftreten, sind die Eingänge automatisch mit dem ALU verbunden. Die Eingangsselektionsschaltung mit den vierfachen UND-ODER-Gattern SN 74157 sorgt dafür. Das Clocksignal kommt von VDRRKL (Multiplikation und Division) oder von ALDSTK (ALU als DST). Die drei genannten Clocksignale (VDRRKL, ALDSTK und R1DSTK/RϕDSTK) sind negative Impulse, im Ruhestand sind die Clocksignale 1. Das UND-Gatter, das die drei Signale verknüpft, hat daher ODER-Funktion!
Die Ausgänge von R_1 und R_ϕ sind mit den (A-)Eingängen des ALU fest verbunden. Ferner werden sie mit den DATA-Leitungen über die bekannte, aus drei SN 7438 bestehende Schaltung verbunden, wenn R_1 oder R_ϕ als SRC adressiert werden. Die Steuersignale sind hierbei R1SRC und RϕSRC.

R_2 und R_3

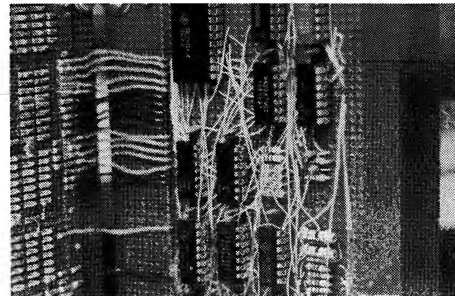
Das Multiplikations-/Divisionsregister R_2 und das Dividier-Overflowregister R_3 bestehen beide aus je drei IC's vom Typ SN 74179: 4 Bit-Parallel/Seriell-Ein, Parallel-Aus-Schieberegistern (Bild 33 und 34). In Kaskade geschaltet bilden sie ein 12 Bit-Schieberegister. Sie besitzen drei Kommandoeingänge: CL (Clock), L (Load) und S (Shift). Abhängig von einer 1 an L oder S wird bei der negativen Flanke des Clockimpulses eingelesen oder geschoben. S ist über L dominant: Wenn S = "1" ist, wird geschoben. Erst wenn S = "0" und L = "1" ist, wird die Information von den Paralleleingängen in das Register übernommen.
R_2 muß als SRC und als DST arbeiten können, R_3 nur als SRC. Über ihre



ALU-Modus-Selektor (ALUMOD).

32

RechenRegister (RR = R_1 & R_ϕ).



Serieneingänge erhalten beide Register Informationen von der Multiplizier-/Dividierschaltung. Der (Serien-)Ausgang von R_2 führt wieder zu dieser Schaltung zurück (und liefert B_i bei der Multiplikation).
Der S-Eingang von R_2 wird beim Multiplizieren und Dividieren (VDDST) 1, der von R_3 nur beim Dividieren (DLDST). Das Clocksignal für beide Register stammt wieder von der Multiplizier-/Dividierschaltung (VDR2KL und DLR3KL); für R_2 im Fall, daß R_2 DST ist, jedoch von der Adress-Selektionsschaltung (R2DSTK).
Die Paralleleingänge von R_3 werden nicht benutzt. Die SN 7400-Serie stellt kein Register zur Verfügung, das ausschließlich serielle Eingänge und parallele Ausgänge besitzt.
In den verschiedenen Schaltungen kommt ein Signal PWRCLR vor. Es handelt sich um ein Resetsignal, das nach Einschalten der Betriebsspannung einen definierten Anfangszustand herstellt (PoWeRCLeaR). Dieses Signal gelangt (als "BUS"-Leitung) zu allen Einheiten, bei denen Zähler, Register usw. zu Beginn gelöscht werden müssen.

Carry-Out- und Gleich-Register

In Bild 35 ist die Schaltung von CO und GL angegeben. Zwei D-Flipflops (SN 7474) erhalten ihr Data-Signal (D) von ALUCO und ALUGL. Die Data-Information wird in das CO-Register eingespeichert, wenn ALU oder GL als DST auftreten (ALDSTK und GLDSTK); GL wird ausschließlich bei einem Vergleich-Auftrag verändert (GLDSTK). In der Funktion als SRC setzen die Flipflops ihre Information, gesteuert von COSRC und GLSRC, auf die DATA-Leitung DATA ϕ1.

Adressenselektor

Zum Schluß soll die Adressen-Selektionsschaltung betrachtet werden. Diese besteht wie allgemein gebräuchlich aus zwei Teilen: der eine für die SRC-, der andere für die DST-Adressen. Als SRC können (über die Selektor-Eingangsgatter) gewählt werden: CO (7767), GL (7766), R_3 (7765), R_2 (7764), R_1 (7763) und $R\phi$ (7762).
Die höheren 9 Bits der Adresse lauten hier 111 111 110 bzw. 776, die niedrigeren 3 Bits bezeichnen eine der genannten Einheiten. Die Selektion übernimmt ein SN 74155, ein doppelter 2 nach 4-Dekoder, verwendet als 3 nach 8-Dekoder (siehe Bild 36).
Auch auf der DST-Seite ist eine solche Schaltung zu finden. Erst werden die höheren 6 Adressen-Bits auf 111 111 (77..) untersucht, dann folgt eine Unterscheidung an Hand der folgenden 2 Bits: Wenn nicht beide 1 sind, handelt es sich um eine ALU-Funktion (7700 ... 7757); sind beide 1 und ist außerdem DSTϕ3 0 (776.), so ist von einem Rechenregister oder von Multiplikation/Division die Rede.
Die erste Gruppe wird mit AL(U), die zweite mit AR (von ARithmetic) bezeichnet. Schließlich werden

33

Multiplizier/Dividier-Register R_2.

34

Overflow-Register R_3.

35

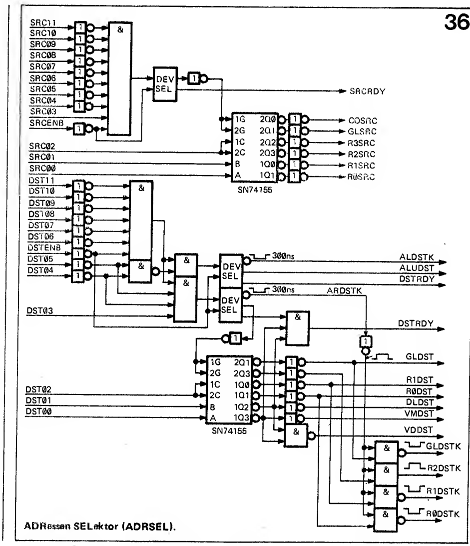
Carry-Out- & Gleich-Register (CO und GL).

im letzten Fall die niedrigsten 3 Bits wieder durch ein SN 74155 ausgewertet, das Signale an GL, R_2, R_1, R_ϕ, DL und VM gibt.

Die DEVice SELector-Schaltungen (in früheren Teilen des Artikels besprochen) liefern impulsförmige Signale, die mit den –DST-Signalen die –DSTK-Signale bilden. Die Polarität dieser impulsförmigen Signale ist so gewählt, daß die aktive Flanke stets am Ende des Impulses liegt. Das –DST-Signal selektiert für jedes Register das Data-Eingangssignal, dann wird nach einer ausreichend langen Verzögerung die Data-Information eingelesen.

Von VMDST und DLDST ist VDDST abgeleitet; es zeigt an, daß eine Multiplikation oder Division stattfindet. Das gleiche Signal hindert die betreffende DEVSEL-Schaltung daran, DSTRDY zu steuern. Bei der Multiplikation/Division wird DSTRDY durch VDDRDY gesteuert (siehe Bild 28).

(wird fortgesetzt)



ADRessen SELektor (ADRSEL).